Київський національний університет імені Тараса Шевченка

Економічний факультет

Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему «ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МАРКЕТИНГОВИХ КАМПАНІЙ В ІТ-ПРОДУКТІ НА ОСНОВІ МОДЕЛЮВАННЯ ДАНИХ ПРО ДОХОДИ КЛІЄНТІВ»

студентки 4 курсу
спеціальності 051 «Економіка»
ОП «Економічна кібернетика»
денної форми навчання
Масник Юлії Олександрівни

Науковий керівний:
доктор економічних наук, професор
Ставицький Андрій Володимирович

Засвідчую, що в цій роботі немає запозичень із праць інших авторів без відповідних посилань

Студент_____
(підпис)

Роботу допущено до захисту перед ЕК
рішенням кафедри економічної кібернетики
від 12 червня 2023 р., протокол №17

Завідувач кафедри:
Доктор економічних наук, професор
Ляшенко Олена Ігорівна _____
(підпис)

КИЇВ – 2023

РЕФЕРАТ

Кваліфікаційна робота бакалавра містить: 104 ст., 47 рис., 13 табл., 44 джерел та додатки

Ключові слова: моделювання доходів клієнтів, маркетингова ефективність, маркетинг в IT-продукті, методи машинного навчання

Об'єкт дослідження: маркетингові кампанії та стратегії IT-продукту

Мета дослідження: моделювання доходу від клієнтів задля підвищення ефективності маркетингової діяльності

Методи дослідження: моделі машинного навчання, зокрема LGBMClassifier, когортний аналіз, статистичні методи

Наукова новизна, теоретична значимість дослідження: змодельовано дохід клієнтів IT-продукту методами машинного навчання в контексті роботи з маркетингом для підвищення ефективності інвестицій по залученню клієнтів

Практична цінність: за допомогою розуміння потенційного доходу від залучених клієнтів процес прийняття рішень щодо оптимізації маркетингових кампаній стає простішим та більш точним, що дає змогу краще інвестувати кошти та досягати поставлених цілей

RESUME

Taras Shevchenko National University of Kyiv,

Faculty of Economics, Department of Economic Cybernetics

Key words: customer revenue data modeling, effectiveness of marketing, marketing in an IT product, machine learning methods

The graduation research describes the process of modeling customer revenue data by using machine learning methods. The resulting data can be useful for increasing effectiveness of marketing in an IT-product

The work is interesting for those people, who are involved in marketing, customer acquisition and the process of increasing return on investment.

Pages 96, pictures 47, tables 13, bibliog. 44, append.

ЗМІСТ

ВСТУП

Сьогодні, коли на ринку існує багато продуктів та послуг, зокрема на IT-ринку, дедалі частіше постає питання не як виробити, а як продати. Так маркетинг набуває нових сенсів та збагачується новими підходами по мірі того, як вдосконалюються технології та змінюються тренди серед суспільства.

В контексті ефективної роботи із просуванням постає питання рентабельності інвестицій та досягнення цілей, які ставляться перед маркетингом, а саме – постійне залучення нових клієнтів, підвищення свідомості та впізнаваності бренду, підвищення доходів, аналіз та вдосконалення маркетингових стратегій.

Останнє в свою чергу обумовлює ***актуальність даного дослідження***, що полягає в покращенні інструментів аналізу та вдосконаленні процесу прийняття рішень шляхом моделювання даних про доходи клієнтів. Це є важливим аспектом при ціноутворенні, впровадженні політики лояльності, сегментації аудиторії, розробці нових продуктів та найважливіше – при стратегічному плануванні та дослідженні питання окупності бізнесу.

Дослідження даного питання мало представлено серед наявних наукових робіт, проте все ж можна проглянути можливі підходи, які застосовуються наприклад в статтях Subhenur Latif та Zakia Zaman Lecturer «Customer annual income prediction using resampling approach» [43], Jing Li, Shuxiao Pan, Lei Huang, Xin Zhu «A machine learning based method for customer behavior prediction» [44]. Зокрема цікавим тут є факт використання методів машинного навчання для досягнення поставлених цілей.

Тож наведені вище аспекти та потреба у вирішенні даної задачі на практиці обумовили вибір такої теми дослідження та сформулювали наступну ***мету дослідження***: змоделювати дохід від клієнтів задля підвищення ефективності маркетингової діяльності.

Об'єкт дослідження: маркетингові кампанії та стратегії IT-продукту.

Предмет дослідження: методи моделювання даних про доходи клієнтів IT-продукту.

Завдання дослідження:

1. Дослідити питання маркетингу в IT-продукті, його складові та можливі підходи до підвищення ефективності;
2. Ознайомитися з підходами до моделювання методами машинного навчання;
3. Моделювання даних про доходи клієнтів на основі моделей машинного навчання.

Методи дослідження: моделі машинного навчання (k-NN, логістична регресія, випадковий ліс та LGBMClassifier), когортний аналіз, статистичні методи, а також методи порівняння та узагальнення.

Наукова новизна роботи: вперше було змодельовано дохід клієнтів IT-продукту методами машинного навчання в контексті роботи з маркетингом для підвищення ефективності інвестицій по залученню клієнтів.

Практична цінність: за допомогою розуміння потенційного доходу від залучених клієнтів процес прийняття рішень щодо оптимізації маркетингових кампаній стає простішим та більш точним, що дає змогу краще інвестувати кошти та досягати поставлених цілей.

Структура роботи: дана робота містить вступ, висновки, три розділи, список джерел, що містить 44 елементи та додатки. Загальний розмір роботи – 104 сторінки, де текст складає 53 сторінки.

РОЗДІЛ 1. РОЛЬ МОДЕЛЮВАННЯ ВИРУЧКИ ДЛЯ МАРКЕТИНГУ В ПРОДУКТОВОМУ ІТ

1.1 Сутність та значення маркетингу в продуктовому ІТ

За даними Мінфін в 2021 рік ВВП України становив 200 млрд. дол [1]. В цей же рік виручка компанії Google становила 256 млрд. дол. При тому якщо розкласти виручку компанії на різні джерела, то доходи від реклами – то 80% або 209 млрд. дол. за рік. Тобто виходить, що реклама, яку просуває окрема платформа, генерує більше грошей, ніж населення окремої країни за аналогічні періоди [2]. Але дане співставлення не несе в собі поганого контексту у вигляді недопрацювання населення – воно лиш чисельно ілюструє сучасні масштаби розвитку маркетингу на базі тільки однієї платформи залучення.

Якщо говорити про маркетинг, як окрему галузь науки, то можна звернутися до визначення даного поняття, наданого її засновником – Ф. Котлером, а саме: «Маркетинг – це не просто продаж, це радше процес надання більшої цінності товару, який надалі братиме участь у взаємовигідному обміні між продавцем та клієнтом» [3].

Основні принципи концепції маркетингу за Ф. Котлером включають [3]:

- Фокус на споживача: маркетингова діяльність повинна бути спрямована на задоволення потреб і бажань споживачів. Важливо розуміти їхній контекст, переваги, цінності та мотивації;
- Обмін цінностями: маркетинг виникає через обмін цінностями між продавцем і споживачем. Цінність може бути матеріальною, інформаційною, соціальною або іншою формою користі для споживача;
- Задоволення потреб: маркетингова стратегія має спрямовуватися на задоволення потреб споживачів. Це означає пропонування продуктів або послуг, які відповідають їхнім потребам і викликають задоволення;
- Взаємовідносини зі споживачами: розуміння, залучення і утримання споживачів є важливим аспектом маркетингу. Взаємодія з клієнтами побудована на взаємовигідних взаємовідносинах, довірі та забезпеченні високої якості обслуговування;

- Управління ринком: маркетингова стратегія має бути орієнтована на управління ринком і конкуренцією. Це включає аналіз ринку, визначення цільових сегментів, розробку продуктів і просування на ринку.

При тому, що хоч Ф. Котлер і вважається батьком сучасного маркетингу, його концепція актуальна як для інтернет-джерел, так і до інших методів просування, що активно використовувались людьми роками. Так якщо розглянути еволюцію застосування різних підходів до просування продуктів, то можна отримати наступну картинку за 1980-2020 роки [4]:

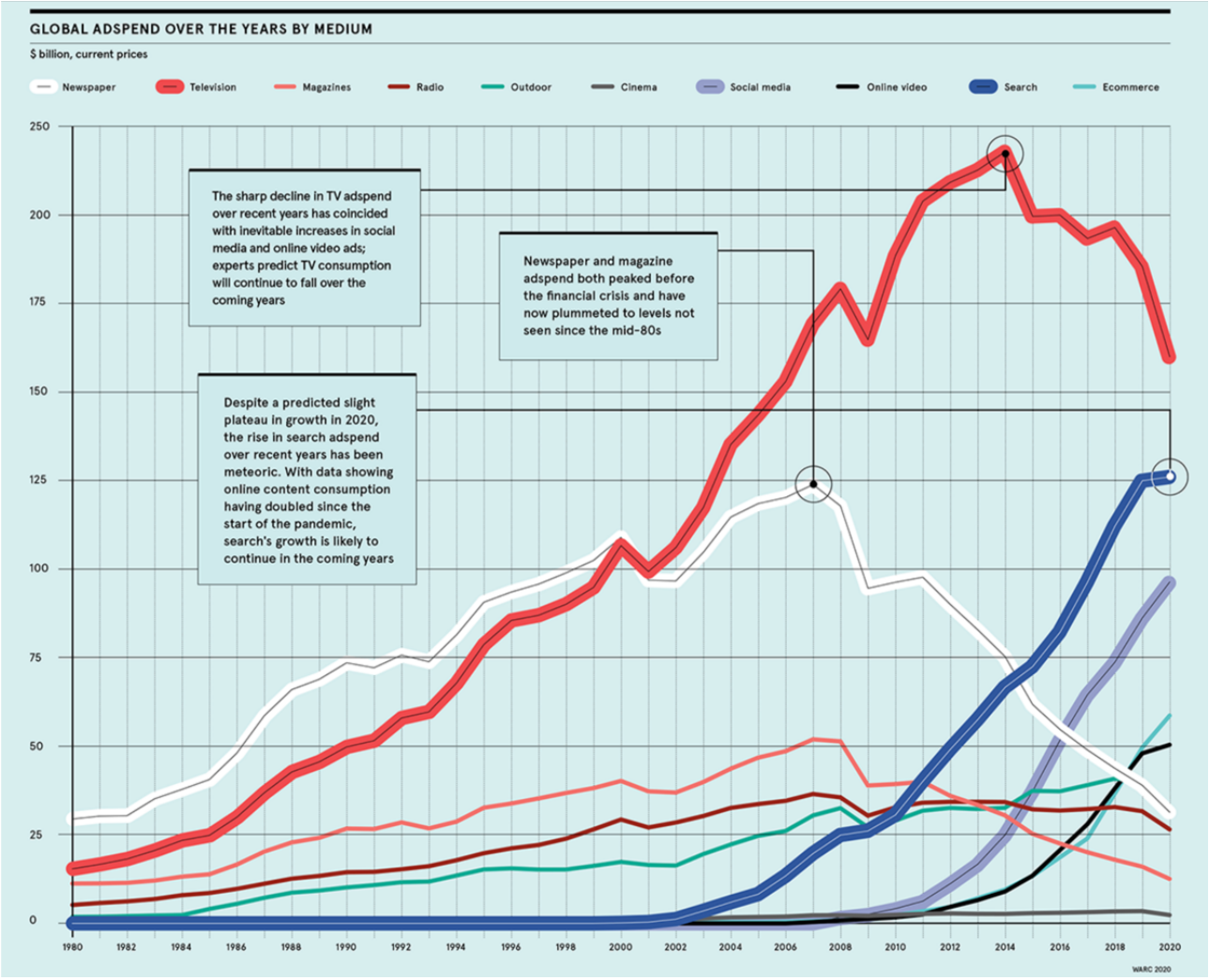


Рис. 1.1. Витрати на рекламу через різні джерела впродовж 1980-2020 років Джерело: [4].

Тут явно виділяється лідерство телевізійного рекламування та просування через газети, але нижче під ними розосереджені по суті різні механіки

просування саме через інтернет, саме те, що називають digital marketing. Узагальнена картинка буде виглядати наступним чином:

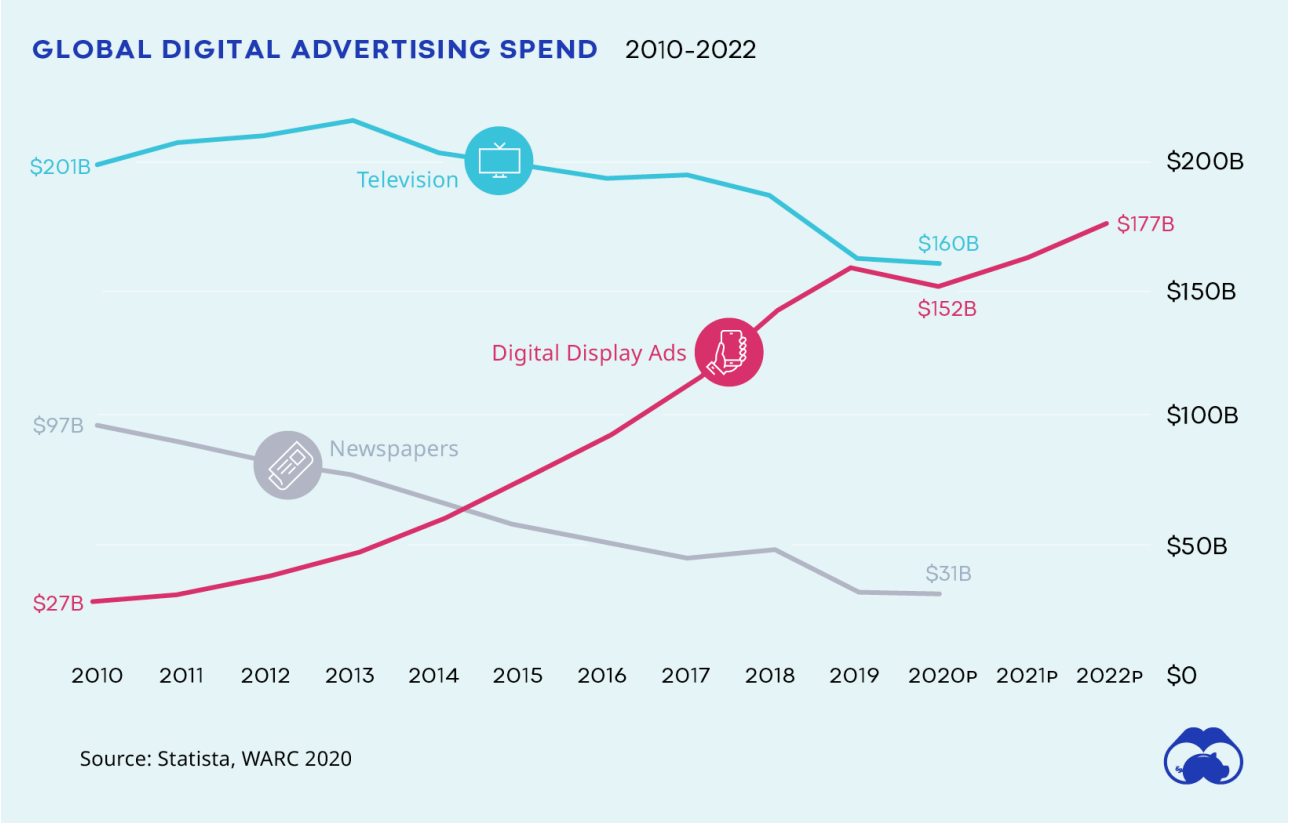


Рис. 1.2. Витрати на рекламу за групами джерел впродовж 2010-2022 років Джерело: [4].

Тобто бачимо, що все рухається в сторону оцифровування, зокрема це може пояснюватись такими його основними перевагами при застосуванні:

- врахування «думки» кожного, персоналізація, виставлення вузького таргету (окремої аудиторії з окремим ознаками);
- активне залучення клієнта в процеси у вигляді перегляду ним рекламних креативів, надання консультацій та проходження анкетування;
- багато доступної інформації і даних для прийняття рішення, за рахунок чого можна тестувати різні гіпотези і робити з того висновки, проводити дослідження ринку та сегментувати аудиторію.

І тут в нагоді стає вже інше висловлення про сучасний маркетинг: «маркетинг – це вміння доносити до користувача здатність закриття його потреби (а іноді навіть і підсвічувати саму потребу), вміння подобатись та викликати довіру» - Д. Янч [5]. Таке визначення важливе в даному контексті, оскільки зі зростанням

ринку інтернет-реклами на ньому зростає і конкуренція, і кількість підходів до просування. Тут виділяються навіть різні зони маркетингу, що за ці різні підходи відповідають та закривають поставлені перед ними цілі, покращуючи при цьому взаємодію з цільовою аудиторією.

Так в продуктовому IT зазвичай виділяють наступні напрямки роботи маркетингу: performance-, product- та brand-marketing.

Основна ідея performance marketing полягає в тому, щоб виміряти і оцінити результати рекламної діяльності, такі як конверсії (момент здійснення цільової дії клієнтом), продажі, генерація лідів (клієнтів з цільовою дією) або інші дії, які є важливими для бізнесу. На відміну від традиційних форм маркетингу, де розміщення реклами може бути оплачено наперед без прямої залежності від результатів, performance marketing спирається на результати та метрики успішності.

Одна з основних переваг **performance marketing** полягає в тому, що вона дозволяє бізнесам зосередитись на конкретних результативних діях та отримувати максимальний рівень повернення від витрат на маркетинг. За допомогою аналізу даних та постійної оптимізації кампаній, компанії можуть ефективно залучати нових клієнтів, збільшувати продажі та досягати своїх бізнес-цілей [6].

Напрямок performance собі зазвичай містить 3 основних вектори роботи:

- ***media buying*** – приведення клієнтів шляхом прямої закупівлі показів реклами на рекламних майданчиках;
- ***affiliate marketing*** – займається приведенням лідів шляхом співпраці з партнерами, що закуповують трафік з рекламних майданчиків. Плата відбувається за конверсію, а не за показ реклами;
- ***influencer marketing*** – покликаний збільшувати кількість цільових дій на основі партнерства з блогерами, плата зазвичай рахується на основі кількості охопленої аудиторії та здійснених ними цільових дій.

Всі ці напрямки працюють в комплексі та допомагають отримувати максимальну вигоду з інвестування коштів, хоча водночас можуть бути

конкурентами один одному, особливо що стосується партнерського трафіку та трафіку, що закуповується напряму через систему аукціону, реалізовану на рекламних майданчиках. Тому тут, як і всюди, потрібен баланс.

Тобто до плюсів в такому виді просування можна віднести вимірюваний результат, можливість контролювати та передбачуваність. До мінусів в свою чергу – залежність від великих партнерів, від платного трафіку та можливі проблеми з прозорістю логік.

В свою чергу **product marketing** - це стратегічна функція в маркетингу, що спрямована на просування та успішний запуск продукту на ринок. Вона орієнтована на розуміння цільової аудиторії, розвиток ефективних маркетингових стратегій та комунікацію цінності продукту для споживачів. Такий підхід є важливим елементом в управлінні, оскільки допомагає зрозуміти та використовувати унікальні особливості продукту для досягнення конкурентної переваги на ринку. Він взаємодіє з іншими функціями маркетингу, такими як маркетингові дослідження, реклама, ціноутворення та продажі, для досягнення загальних цілей бізнесу [7].

До інструментів роботи тут можна віднести аналіз ринку та конкурентів, стратегію виходу на ринок, контент маркетинг, робота зі сповіщеннями через пуші та електронну пошту тощо.

До плюсів такого підходу можна віднести високу ефективність, гнучкість та довгостроковий ефект, до мінусів – потреба у великих обсягах залученої аудиторії та в додатковому залученні ресурсів розробки та сторонніх сервісів (як у випадку роботи зі сповіщеннями).

Якщо говорити про історію з брендом, що виражається через **brand-marketing,** варто зазначити, що це стратегічна галузь маркетингу, яка зосереджена на розвитку, управлінні та просуванні брендів. Вона спрямована на створення сильних і унікальних брендів, які мають визнання, цінність та відмінність на ринку [8].

До векторів роботи в даній сфері зазвичай відносять роботу з SMM, PR, партнерствами, інфлуенсерами та безпосередньо розвиток самого бренду. Такий

підхід надає конкурентні переваги в довгостроковому періоді, однак проблемою водночас може стати важкість виміру ефективності та потребу в довгострокових інвестиціях.

Тобто в підсумку можна сказати, що напрямок бренду вимагає інвестиції, окупність яких важко виміряти, бо зазвичай результат тому більша впізнаваність та приріст в органічному трафіку (органічний – той, на який бізнес не витрачає гроші). Напрямок продуктового маркетингу натомість менш затратний, навіть умовно безкоштовний і дає ефект в залученості користувачів та покращенню їх досвіду користування продуктом більш в якісному плані. А performance-marketing в собі включає чіткі значення інвестицій та майже чіткі значення по виручці, що прийде з тих інвестицій. Чому майже чіткі знання по виручці – тому що з самого початку, коли треба оцінювати ефективність витрат станом на зараз, важко сказати, скільки ми заробимо в майбутньому. Тому звідси і виникає більш детальна потреба в роботі з моделюванням доходу для кращого масштабування роботи безпосередньо з ефективністю. Але спочатку слід більш детально ознайомитись з підходами в performance-marketing, а саме інструментами, за допомогою яких обчислюється результативність маркетингової діяльності компанії.

1.2 Роль змодельованого доходу при зведенні юніт-економіки та формуванні цілей по окупності

У постійному процесі інвестування коштів сформувалась своєрідна умова успішності будь-якого бізнесу – знаходження unit-економіки. Саме її наявність або відсутність є індикатором знаходження свого місця на ринку продуктом, вказівником про можливість отримувати прибуток та залучати подальші інвестиції.

Unit-економіка (англ. unit economics) - це фінансовий аналіз, що оцінює дохід, витрати та прибутковість одиниці продукції або послуги, яку надає компанія. В основі unit-економіки лежить принцип розгляду бізнесу на рівні окремих одиниць, які генерують дохід. Тобто при застосуванні даного підходу на практиці ми отримуємо розуміння, на скільки ефективно інвестується 1$ та чи заробляємо ми достатньо з 1 клієнта або 1 покупки [9].

Показником знаходження юніт-економіки є величина співвідношення між доходом від клієнта за час користування продуктом (lifetime value або LTV) та витратами на його залучення (customer acquisition costs або CAC). Для масштабування ідеальним є варіант LTV:CAC=3:1, проте навіть факт того, що LTV>CAC уже говорить про те, що бізнес не є збитковим [10].

Якщо говорити детальніше про обчислення, то для розрахунку CAC потрібно враховувати загальну суму витрат, пов'язаних з маркетинговими активностями, просуванням продукту, продажами та іншими затратами, які сприяють залученню нових клієнтів [11].

Формула для розрахунку CAC в собі містить всі затрати поділені на всіх залучених клієнтів. Наприклад, якщо компанія витратила $10000 на маркетингові активності, а в результаті залучила 100 нових клієнтів, CAC складатиме $100 ($10000 / 100).

Чим більше цільових клієнтів залучається, тобто чим краще налаштована маркетингова діяльність – тим дешевшою є закупка і тим більшою може бути прибуток від кожного юніта.

Для розрахунку LTV в свою чергу потрібно враховувати середній час, протягом якого клієнт залишається клієнтом компанії (тривалість життя), середню вартість покупок або прибутку, який компанія отримує від клієнта за певний період, а також прибуток, який компанія очікує отримати від клієнта в майбутньому [11].

Формула для розрахунку LTV може мати різні варіації, але загальний підхід включає в себе знання про час життя на продукті клієнта та його середній чек. Наприклад, якщо середня вартість покупок клієнта на рік складає $500, а відомо, що середня тривалість життя клієнта складає 5 років, LTV складатиме $2,500 ($500 * 5).

Але якщо витрати ми знаємо і можемо порахувати відразу, то частина з потенційними надходженнями від того чи іншого клієнта невідома. А це знання критично необхідне для розуміння ефективної роботи маркетингу. Тут виникає потреба в додатковому аналізі та моделюванні.

Далі, якщо відходити від поняття юніта та говорити про цілепокладання загалом, то цей процес на практиці виглядатиме наступним чином. Припустимо є країна, де залучаються клієнти, і є розуміння на основі історичних даних, що через 3 місяці витрачені гроші повернуться, настане момент окупності і, відповідно, момент, коли можна реінвестувати, що дуже бажано для бізнесу.

Момент окупності тут – це невід’ємне значення ROI (return on investment). При тому, коли ми говоримо про ROI – сюди включаються і витрати на залучення клієнтів, і комісії, які платяться за них, повернення коштів тощо. Часто в контексті маркетингу під такою окупністю мають на увазі ROMI (return on marketing investment), оскільки на цьому етапі у витрати не закладаються відрахування на заробітні плати, оренди приміщень і т.д.. Також в контексті залучення є й інший вид окупності – ROAS (return on ad spend) – який в собі враховує витрати лише по рекламуванню. Всі ці види рентабельності рахуються та є важливими на різних етапах оцінки результативності інвестицій [12].

Далі, маючи знання про клієнтів з повною окупністю на 3 місяць, ми можемо проглянути, якою була маркетингова окупність у таких клієнтів на даній локації

станом на перший місяць залучення та, відповідно до цього відсотка, поставити ціль по залученню нових клієнтів для маркетингових команд. В даному випадку при виставленні цілі ми переходимо від оцінки 3 місяців до одного, оскільки за допомогою зібраних раніше даних можемо розрахувати коефіцієнти «дотікання» виручки за наступні 2 місяці, маючи інформацію лише по першому місяцю, що знову ж таки пришвидшує процес оцінки. І, як результат такого процесу, маємо наприклад ціль по залученню клієнтів певної країни у вигляді -40% ROMI в перший місяць, аби через 3 місяці почати заробляти та реінвестувати.

Але в підході, описаному вище, жодним чином не згадувалось прогнозування виручки, проте тут цей процес стає важливим саме після моменту знаходження цих -40% цілі, а саме:

- для того, щоб на початку місяця розуміти, чи буде досягнута ціль в кінці місяця, ми прогнозуємо показник окупності і спостерігаємо за його значенням на щоденній онові. Якщо бачимо, що потрібне нам значення не досягається – дана маркетингова кампанія зупиняється та перезапускається;
- оскільки ми плануємо покладатися на значення прогнозних даних, при виставленні фінальних цілей на команду необхідно закладати похибку, отриману при моделюванні, і якщо ми розуміємо, що це 5 п.п. - ми закладаємо це в обрахунках та даємо цільовий інтервал 35-45% для подальшого процесу прийняття рішення щодо ефективності закупки.

Тобто бачимо, що в той час, коли основна ціль продукту – заробити гроші, ціль маркетингу – правильно приймати рішення щодо інвестицій часто в умовах невизначеності аби забезпечити можливість продукту заробити гроші. Тому часто вирішальним фактором успіху на етапі залучення є правильно вибудувана система аналітики та прийняття рішень на основі фактичних та змодельованих даних.

1.3 Перевірка гіпотез через A/B-тестування та оцінка фінансових результатів

Швидкий розвиток IT-індустрії та постійні зміни на ринку вимагають від компаній постійно покращувати свої продукти. Один з ефективних способів, що забезпечує це постійне покращення - це A/B тестування. Тестування зі співставленням контрольної та тестової версій дозволяє провести експеримент з альтернативними варіантами лендингу (сторінки, на яку потрапляє користувач після кліку по рекламі), сайту або додатку, щоб визначити, який з них більш краще сприймається користувачами та відповідно більш прибутковий для компанії [13].

Основні переваги, які можна отримати від проведення A/B тестування:

- ***Покращення користувацького досвіду:*** експерименти з різними варіантами дизайну та функціональності можуть підвищити рівень задоволення від використання продукту та збільшити лояльність користувачів.
- ***Зростання конверсії:*** тестування може покращити розуміння змін, що впливають на відсоток користувачів з цільовою дією, - реєстрацією, покупкою тощо – та надалі масштабувати ті з них, що дали приріст.
- ***Ефективність витрат:*** перевірка життєздатності ідей з поділом на групи зазвичай 50% на 50% дозволяє визначити, чи дійсно суб'єктивні зміни від команди покращують наявні метрики, перед тим, як імплементувати такі оновлення для всіх користувачів і втрачати гроші у 100% випадках, а не лише в 50%.
- ***Оптимізація продукту:*** внаслідок постійної перевірки різного роду гіпотез можна краще розуміти потреби та болі своєї аудиторії та покращувати наявний функціонал продукту, його ефективність, іноді навіть шляхом регресивного тестування.

Отже наявність налаштованого процесу перевірки гіпотез та велика кількість тестів, що приносять релевантні результати якомога швидше, – запорука масштабування бізнесу.

Далі детальніше про специфіку тестів в маркетингу: під час роботи з рекламою на різних джерелах залучення трафіку оплата стягується на етапі показу креативу (рекламного оголошення) – тобто верх усієї воронки залучення. Далі при кліку на рекламу клієнт потрапляє на ленд, основне завдання якого розігріти, підтвердити очікування від побаченого в рекламному оголошенні та сприяти перш за все реєстрації та подальшій оплаті вже в рамках користування продуктом. Тобто в процесі залучення важливо не лише проводити експерименти з дизайном та копірайтом креативів, а й покращувати прохідність кроків лендингів та їх якість.

Так якщо команда Media Buying приводить на лендинг одного користувача за 1$, то далі дуже бажано, аби вартість доведення потенційного клієнта до цільової дії збільшилась мінімально. Тобто ще на етапі показу реклами для кліку по ній від тисячі користувачів ми заплатимо 1 000$. Далі при конверсії в реєстрацію у розмірі 10% ми приведемо на продукт 100 користувачів за ціною 10$. При збільшенні відсотку конверсії принаймні на 1 п.п., зареєстрованих буде вже 110 клієнтів із вартістю 9,09$. Бачимо, що чим більша прохідність воронки залучення – тим дешевше вартує залучення і, відповідно, швидше наступає окупність та збільшується маржинальність. Тобто в підсумку сказаного вище – основна метрика тут – конверсія в реєстрацію.

Але насправді не менш важливо надалі дивитися на платежі від залучених клієнтів, їх взаємодію з продуктом і бажанням повертатися і платити вдруге та втретє, оскільки цілком реальною є можливість зробити реєстрацію в один клік з конверсією майже 100%, але це не свідчитиме про приведення якісних клієнтів і швидку їх окупність, приведених умовно за 1$. Тобто тут виникає необхідність в орієнтуванні не лише на відсоток реєстрацій від усіх, хто потрапив на лендинг, а й на конверсію в покупця, середній чек та загалом згенерований дохід від клієнтів.

На практиці описаний вище процес аналізу виглядає наступним чином: на перший екран лендингу внесли зміни в тексті, внаслідок чого були отримані результати, зображені на рис.1.3.

	Control (0) 137mf adaptive	Test (1) 137mf adaptive
Unique Visits	74 553	74 459
Visits	105 120	104 087
Registrations	9 055	9 505
Activations	9 055	9 505
CVR, %	8,61%	9,13%
% Difference in CVR		6%
% buyers	1,34%	1,55%
% Difference in % buyers		16%
Payments	250	295
% Difference in ARPPU		1%
Confirmed emails	2 950	3 129
% confirmed	32,58%	32,92%
% Difference in % confirmed		1%
% photo uploaded	53,80%	53,76%
% Difference in % photo upload..		0%
% retention to 2nd day	10,31%	10,54%
% Difference in % retention to 2..		2%
% retention to 2nd session	52,28%	52,82%
% Difference in % retention to 2..		1%

Рис. 1.3. Результати, отримані внаслідок проведення A/B-тестування

Джерело: складено автором на основі даних компанії.

Метрик виведено вдосталь для перевірки різних аспектів впливу тесту, однак основні з них, на які звернемо увагу, – це %CVR (конверсія в реєстрацію), %buyers (конверсія в платника) та ARPPU (середній чек платника).

Так при оцінці статистичної значущості отриманої різниці в конверсії в реєстрацію маємо ситуацію, де підтверджується, що отриманий приріст у розмірі +6% буде зберігатися більш ніж в 99% ситуацій і отриманий результат не є випадковістю – рис.1.4.

Водночас при аналогічному оцінюванні позитивних змін у конверсії в платника отримали ситуацію, коли дані зміни не можна назвати статистично значущими – рис.1.5.

Якщо говорити про оцінку того, скільки платять клієнти – отримуємо знову ж таки ситуацію невизначеності з боку підтвердження статистичної значущості отриманої різниці – рис.1.6.

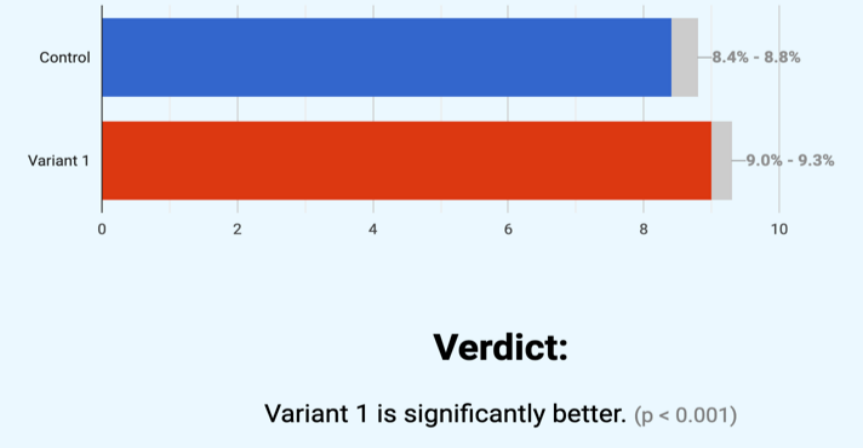


Рис. 1.4. Перевірка статистичної значущості різниці в конверсії в реєстрацію

Джерело: складено автором на основі даних компанії.

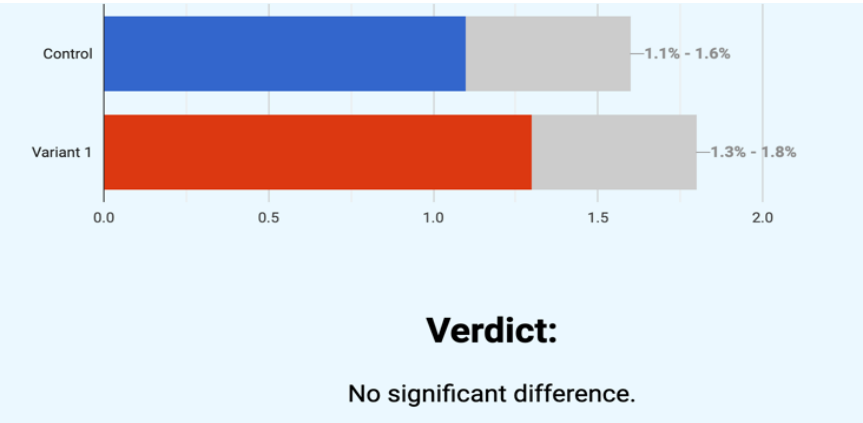


Рис. 1.5. Перевірка статистичної значущості різниці в конверсії в платника

Джерело: складено автором на основі даних компанії.

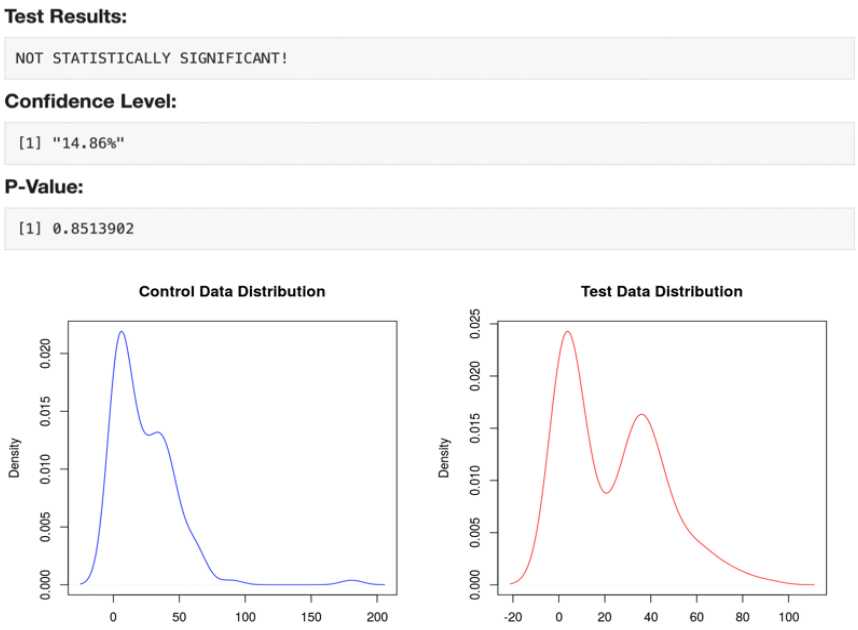


Рис. 1.6. Перевірка статистичної значущості різниці в середньому чеку.

Джерело: складено автором на основі даних компанії.

І тут спостерігаємо проблему: зазвичай відвідувачів сайту та здійснених реєстрацій на ньому вистачає для того, аби обрахувати статистичну значущість та статистичну потужність отриманої різниці на користь тестової групи при проведенні тесту, тим паче тут маємо біноміальний розподіл, де однаково можливі настання або ненастання цільової дії користувачем. Далі конверсія в покупця теж має 2 можливі результати – ти стаєш, або не стаєш платником, тобто теж біноміальний розподіл. Однак у випадку проведення тесту на окремій аудиторії, окремого віку, статі, країни, джерела трафіку – потрібні для аналізу дані зазвичай збираються надто довго. До того ж, більша конверсія в платника, яку ще є ймовірність дочекатися на перших етапах закриття тесту, не завжди говорить про більшу генерацію виручки даною групою клієнтів: так, якщо тест орієнтований приміром на продаж продукту за нижчими цінами, більше людей купуватимуть, але середній чек від того буде меншим. Тому для оцінки повної картини по проведеному тесту варто враховувати, скільки саме грошей принесуть залучені клієнти та чи дійсно наші зміни під час тестування гіпотези принесли бажані результати у фінансовому плані. Однак тут виникає проблема у вигляді того, що розподіл грошей не біноміальний, а радше такий, що складається з неперервних величин, зазвичай експоненціальний, де багато клієнтів платять мало і навпаки.

При тому, що на етапі залучення ми ще не знаємо, скільки клієнтів платитиме, з якої частотою та з яким середнім чеком, завдання оцінки результативності тесту виглядає не вирішеним до кінця при підрахунку лише значущих змін в конверсії в реєстрацію. Звідси і виникає посилення потреби у моделюванні доходу клієнтів для більш точного обрахунку фінансового приросту від втілення тої чи іншої ідеї в життя.

Висновки до розділу 1

З кожним роком маркетинг в інтернеті набирає обертів, маючи в своєму інструментарії різні підходи до роботи з клієнтами з різною кінцевою ціллю. Так performance-, product- та brand-marketing в комплексі дають бажаний результат у

вигляді вигідного залучення якісних клієнтів на відомий продукт з брендом. Однак найбільше переваг щодо вимірюваності результативності має в собі саме performance-marketing, в завдання якого входить створення успішної стратегії маркетингових інвестицій задля отримання бажаного рівня прибутку та моніторинг виконання цілей по окупності.

Так, основною ціллю по окупності для всіх без винятку бізнесів є зведена unit-економіка у вигляді LTV>CAC, бо саме виконання даної умови свідчить про те, що продукт більше заробляє, ніж витрачає. Додатково, якщо в наявності є знання про витрати та доходи від клієнтів, – можна в рамках окремої їх групи, наприклад країни, прорахувати %ROMI на n-місяць, що надалі стане індикатором успішності для маркетингових команд.

До того ж, проводячи регулярні процеси по покращенню методів залучення у вигляді A/B-тестів, команди орієнтуються не лише на здешевлення вартості приведення клієнта на продукт, а й на збільшення фінансових показників. І тут часто для коректної оцінки може бракувати цілісної картини про доходи, що, як і в попередньому пункті з цілепокладанням, може вирішуватись шляхом моделювання грошей, які компанія заробить у майбутньому.

РОЗДІЛ 2. ОГЛЯД МЕТОДІВ МОДЕЛЮВАННЯ ДОХОДУ

2.1 Когортний аналіз доходу від клієнтів

Одним із найбільш популярних і найбільш застосовуваних методів при роботі з даними в IT-продуктах є когортний аналіз, оскільки він дозволяє групувати клієнтів в залежності від певних характеристик або подій. У контексті бізнесу, когортний аналіз використовується для спостереження та аналізу поведінки групи людей, які поділилися спільними характеристиками або здійснили певну подію в однаковий період часу. Зазвичай цей однаковий період часу – це час залучення користувача на продукт: день, тиждень, місяць тощо.

Такий підхід дозволяє виявити тенденції, патерни та зміни в поведінці групи клієнтів з плином часу, при чому ми врівноважуємо цих клієнтів в можливостях принести нам ту чи іншу суму грошей. Так в житті часто говорять: «раніше студенти вчились краще, он яких результатів вже досягли, на відміну від поточних». Проте це викривлення, оскільки студенти 2010 року, на відміну від 2020, мали в запасі додаткових 10 років для своїх здобутків. Тому важливо такий сегмент «студентів» розбивати на групи за роками випуску – тобто когорти - і порівнювати їх в розрізі 1 рік після випуску, 2 роки, 3 роки тощо [14].

До переваг такого підходу в аналізі можна віднести виявлення причинно-наслідкових зв’язків, а саме: продукт покращив рівень повернення користувачів у 1 день – як наслідок нові когорти, в порівнянні зі старими, мають вже не 2% цільової метрики в перший день життя, а 3%.

Також можна краще розуміти проблемні місця на продукті та покращувати їх на нових когортах клієнтів – приміром підвищувати середній чек 3 дня, якщо поточне його значення зараз невигідне бізнесу.

В контексті розглянутого раніше А/Б тестування так можна розуміти в довгостроковій перспективі наслідки внесених змін та правильність прийнятого рішення. Так когорта, що була залучена в рамках тесту, хоч і могла мати значно кращі показники в реєстрації внаслідок нової воронки-квіза, проте далі фінансові метрики можуть мати гірші значення станом на n-день, в порівнянні зі старими клієнтами.

В контексті моделювання доходу варто озвучити ще одну перевагу даного методу, а саме розуміння поведінки клієнтів з плином часу. Коли ми знаємо як поводили себе попередники залежно від дня користування продуктом – можемо застосувати аналогічні оцифровані патерни поведінки по відношенню до нових клієнтів.

В наступному прикладі розглянемо залучених в різні тижні клієнтів січня-березня (coh_week) та дохід, що ті генерують по перших днях життя на продукті (day 0-6). Завдання: на основі історичних даних порахувати дохід від когорти 26.03.23 за перший тиждень життя на продукті, щоб зрозуміти потенціал її залучення надалі. Когорти і дані по їх доходам можна проглянути на рис.2.1.

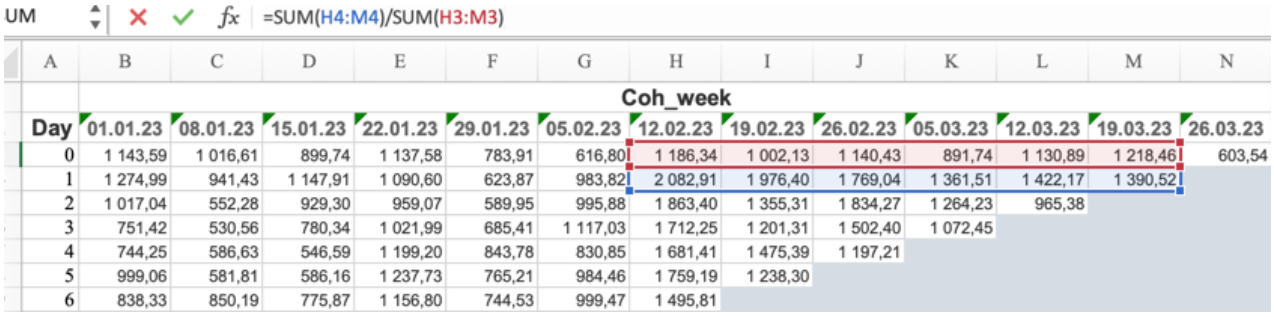
	Coh_week												
Day	01.01.23	08.01.23	15.01.23	22.01.23	29.01.23	05.02.23	12.02.23	19.02.23	26.02.23	05.03.23	12.03.23	19.03.23	26.03.23
0	1 143,59	1 016,61	899,74	1 137,58	783,91	616,80	1 186,34	1 002,13	1 140,43	891,74	1 130,89	1 218,46	603,54
1	1 274,99	941,43	1 147,91	1 090,60	623,87	983,82	2 082,91	1 976,40	1 769,04	1 361,51	1 422,17	1 390,52	
2	1 017,04	552,28	929,30	959,07	589,95	995,88	1 863,40	1 355,31	1 834,27	1 264,23	965,38		
3	751,42	530,56	780,34	1 021,99	685,41	1 117,03	1 712,25	1 201,31	1 502,40	1 072,45			
4	744,25	586,63	546,59	1 199,20	843,78	830,85	1 681,41	1 475,39	1 197,21				
5	999,06	581,81	586,16	1 237,73	765,21	984,46	1 759,19	1 238,30					
6	838,33	850,19	775,87	1 156,80	744,53	999,47	1 495,81						

Рис. 2.1. Згруповані дані по доходах клієнтів за тижневими когортами.
Джерело: складено автором на основі даних компанії.

Так можемо порахувати формування доходу з дня на день на основі використання ланцюгових коефіцієнтів. Тобто для знаходження майбутньої виручки за 1 день, необхідно порахувати, яка її частка переходить з 0 дня на прикладі старіших когорт, і тут постає питання: яке «вікно» даних для того обрати. Якщо проводити обрахунки лиш на останній повністю відомій когорті за визначені дати – 12.02, існує ризик отримати не зовсім реалістичну оцінку ситуації, оскільки так ми не беремо до уваги зміни в продукті, які могли вплинути на ситуацію для новіших когорт. І навпаки, якщо вести обрахунки лиш орієнтуючись на попередню когорту кожного разу – можемо не правдиво завищити/занизити результати, нівелюючи при цьому закладені історично тенденції поведінки.

Таким чином було обрано «вікно» у розмірі 6 когорт від останньої для обрахунку коефіцієнтів переходу, що дозволить як врахувати сучасний стан справ на продукті, так і підстрахуватися давнішими значеннями.

Так обрахунок коефіцієнта 1 дня включає в себе обрахунок частки суми першого дня до нульового в останніх 6 когортах, процес зображений на рис.2.2. Для наступних днів обрахунки йдуть аналогічно зі зміщенням на одну старішу когорту.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
		Coh_week												
	Day	01.01.23	08.01.23	15.01.23	22.01.23	29.01.23	05.02.23	12.02.23	19.02.23	26.02.23	05.03.23	12.03.23	19.03.23	26.03.23
0	0	1 143,59	1 016,61	899,74	1 137,58	783,91	616,80	1 186,34	1 002,13	1 140,43	891,74	1 130,89	1 218,46	603,54
1	1	1 274,99	941,43	1 147,91	1 090,60	623,87	983,82	2 082,91	1 976,40	1 769,04	1 361,51	1 422,17	1 390,52	
2	2	1 017,04	552,28	929,30	959,07	589,95	995,88	1 863,40	1 355,31	1 834,27	1 264,23	965,38		
3	3	751,42	530,56	780,34	1 021,99	685,41	1 117,03	1 712,25	1 201,31	1 502,40	1 072,45			
4	4	744,25	586,63	546,59	1 199,20	843,78	830,85	1 681,41	1 475,39	1 197,21				
5	5	999,06	581,81	586,16	1 237,73	765,21	984,46	1 759,19	1 238,30					
6	6	838,33	850,19	775,87	1 156,80	744,53	999,47	1 495,81						

Рис. 2.2. Процес обрахунку коефіцієнту переходу доходу з 0 в 1 день.

Джерело: складено автором на основі даних компанії.

Отримані коефіцієнти для кожного нового дня на продукті можна проглянути в таблиці 2.1.

Таблиця 2.1

Коефіцієнти переходу доходу між днями користування продуктом

Коефіцієнти	Перехід
1,52	*0-1*
0,86	*1-2*
0,92	*2-3*
1,00	*3-4*
1,00	*4-5*
1,02	*5-6*

Джерело: розрахунки автора на основі даних компанії.

Але варто зазначити, що отримані значення необхідно попередньо згладити, що допоможе знизити волатильність коефіцієнтів і нівелювати можливі викиди в даних. Так отримані показники переходу були згладжені експоненціально з alpha=0,7. Даний вид згладжування був обраний, оскільки коефіцієнти мають гладкий та ніби горизонтальний тренд. Значення константи наближається більше до 1, аби збільшити вагу саме на останні значення часового ряду [15]. Візуалізовані твердження вище можна розглянути на рис.2.3.

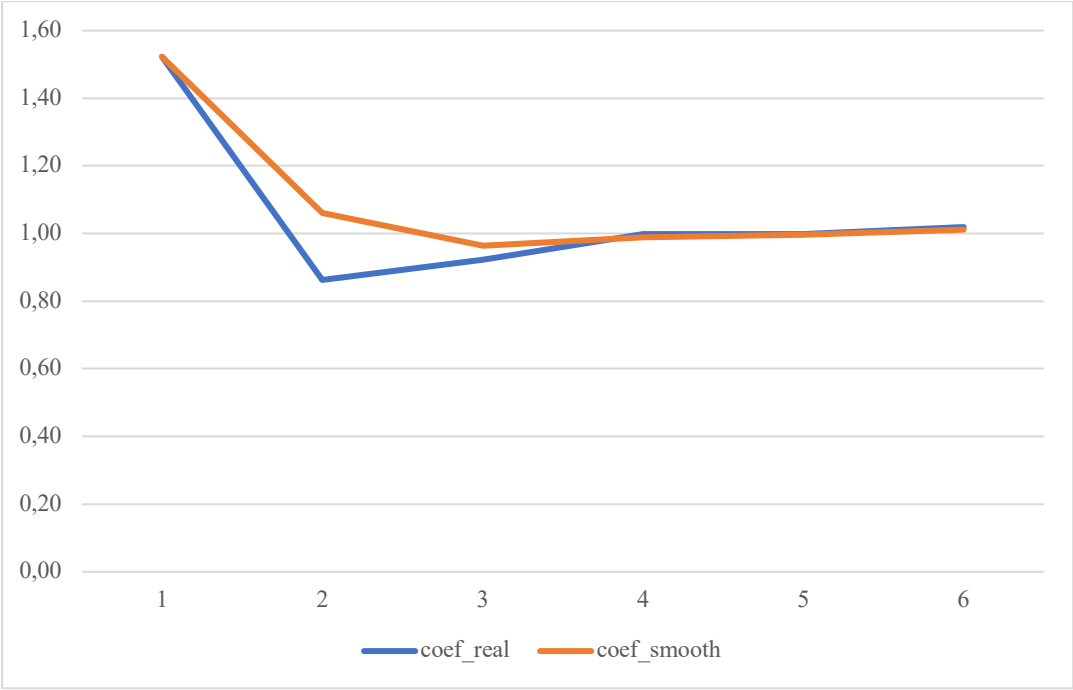


Рис. 2.3. Порівняння «сирих» та згладжених коефіцієнтів переходу

Джерело: складено автором на основі даних компанії.

Так на основі отриманих згладжених значень змогли дозаповнити пропуски в нових когортах і отримати їх прогнозний дохід – рис.2.4-рис.2.5.

M =N3*P3

														O		
	Coh_week															
Day	01.01.23	08.01.23	15.01.23	22.01.23	29.01.23	05.02.23	12.02.23	19.02.23	26.02.23	05.03.23	12.03.23	19.03.23	26.03.23		Коефіцієнти	Перехід
0	1 143,59	1 016,61	899,74	1 137,58	783,91	616,80	1 186,34	1 002,13	1 140,43	891,74	1 130,89	1 218,46	603,54		1,52	*0-1*
1	1 274,99	941,43	1 147,91	1 090,60	623,87	983,82	2 082,91	1 976,40	1 769,04	1 361,51	1 422,17	1 390,52	=N3*P3		1,06	*1-2*
2	1 017,04	552,28	929,30	959,07	589,95	995,88	1 863,40	1 355,31	1 834,27	1 264,23	965,38	1474,84	974,58		0,96	*2-3*
3	751,42	530,56	780,34	1 021,99	685,41	1 117,03	1 712,25	1 201,31	1 502,40	1 072,45	930,60	1421,70	939,47		0,99	*3-4*
4	744,25	586,63	546,59	1 199,20	843,78	830,85	1 681,41	1 475,39	1 197,21	1059,56	919,41	1404,60	928,17		1,00	*4-5*
5	999,06	581,81	586,16	1 237,73	765,21	984,46	1 759,19	1 238,30	1192,11	1055,04	915,49	1398,61	924,22		1,01	*5-6*
6	838,33	850,19	775,87	1 156,80	744,53	999,47	1 495,81	1252,56	1205,83	1067,19	926,03	1414,72	934,86			
Total	6 768,68	5 059,51	5 665,91	7 802,97	5 036,65	6 528,30	11 781,31	9 501,40	9 841,29	7 771,71	7 209,96	9 723,45	6 223,70			

Рис. 2.4. Процес розрахунку майбутнього доходу нових когорт на основі отриманих коефіцієнтів переходу

Джерело: складено автором на основі даних компанії.

	Coh_week												
Day	01.01.23	08.01.23	15.01.23	22.01.23	29.01.23	05.02.23	12.02.23	19.02.23	26.02.23	05.03.23	12.03.23	19.03.23	26.03.23
0	1 143,59	1 016,61	899,74	1 137,58	783,91	616,80	1 186,34	1 002,13	1 140,43	891,74	1 130,89	1 218,46	603,54
1	1 274,99	941,43	1 147,91	1 090,60	623,87	983,82	2 082,91	1 976,40	1 769,04	1 361,51	1 422,17	1 390,52	918,87
2	1 017,04	552,28	929,30	959,07	589,95	995,88	1 863,40	1 355,31	1 834,27	1 264,23	965,38	1474,84	974,58
3	751,42	530,56	780,34	1 021,99	685,41	1 117,03	1 712,25	1 201,31	1 502,40	1 072,45	930,60	1421,70	939,47
4	744,25	586,63	546,59	1 199,20	843,78	830,85	1 681,41	1 475,39	1 197,21	1059,56	919,41	1404,60	928,17
5	999,06	581,81	586,16	1 237,73	765,21	984,46	1 759,19	1 238,30	1192,11	1055,04	915,49	1398,61	924,22
6	838,33	850,19	775,87	1 156,80	744,53	999,47	1 495,81	1252,56	1205,83	1067,19	926,03	1414,72	934,86
Total	6 768,68	5 059,51	5 665,91	7 802,97	5 036,65	6 528,30	11 781,31	9 501,40	9 841,29	7 771,71	7 209,96	9 723,45	6 223,70

Рис. 2.5. Результати когортного моделювання доходу

Джерело: складено автором на основі даних компанії.

Так з результатів бачимо, що користувачі, залучені від 26 березня прогнозно приноситимуть менше виручки ніж старіші когорти – отже варто переглянути способи залучення клієнтів від обраної дати та зміни, які впроваджувались на продукті. Однак дипломна на цьому не закінчується, оскільки є ряд недоліків в такому підході, а саме:

1. якщо розділити дані відносно джерел залучення, статі, віку, країни користувача і будувати дохід за коефіцієнтами більш деталізовано – отримаємо не 6 223 гр. од. в сумі по когорті 26.03, а приблизно 8 тисяч, що більше відповідає реальності. Тому з деталізацією ми не зупинимо помилково трафік, який раніше вважали менш прибутковим. Але таке розбиття – це додаткові таблиці та зменшення вхідних даних до аналізу;
2. такий аналіз може вимагати значної кількості даних, особливо якщо розглядається багато когорт або довгі періоди часу. Це може бути викликом для бізнесів, які не мають достатньої бази спостережень для проведення аналізу;
3. інтерпретація результатів когортного аналізу може бути складною, особливо якщо виявляється багато різних залежностей та трендів. Розуміння, які фактори мають суттєвий вплив на поведінку клієнтів, може бути викликом. До того ж, в прикладі вище ми отримали загальну суму доходу, але так і не дізналися, що впливає на його формування, та хто саме з клієнтів платить.

Озвучені недоліки є досить значними при роботі з прогнозуванням виручки – саме тому є потреба в іншому способі вирішення поставленого завдання. Як варіант – скористатися методами машинного навчання, про що говоритимемо далі.

2.2 Дослідження окремих методів машинного навчання та способів їх оцінювання

При прогнозуванні доходу від клієнтів на продукті з наявністю підписки основним завданням є правильно змоделювати клієнтів, що стануть платниками. Тобто необхідно провести обчислення ймовірності приналежності об'єкту класифікації до певного класу та, на основі даної ймовірності, віднести даний об'єкт до класів «буде платити» чи «не буде платити». Далі з цим знанням та знанням ціни підписки можемо отримати суму доходу і порівняти, на скільки прогнозна її величина відрізняється від реально отриманої.

Відповідно до озвученого вище завдання далі будемо розглядати моделі машинного навчання, які здатні на основі вхідних ознак видавати ймовірність, з якою можна очікувати на платіж від користувача.

Так одним з найпростіших алгоритмів машинного навчання є метод **k-найближчих сусідів** (k-NN), суть якого полягає в розташуванні кожного клієнта відповідно до його ознак ніби в площині і далі на основі голосування заданої наперед кількості найближчих сусідів вирішується, на кого з них більше схожий поточний зразок даних і який клас присвоїти в даному випадку. Банально алгоритм дії тут ілюструється на основі приказки «розкажіть про своїх друзів (сусідів) і я скажу, хто ви». При чому близькість сусідів тут обчислюється різними методами, зазвичай використовується Евклідова відстань [16].

До переваг даного методу можна віднести простоту реалізації, гнучкість, обробка нелінійних залежностей та відсутність потреби у навчанні (k-NN є зразковим алгоритмом, що означає, що він не вимагає тренування моделі на тренувальному наборі даних, весь процес полягає в збереженні тренувальних даних та їх використанні для класифікації або прогнозування нових зразків). Водночас проблемою при використанні може стати обчислювальна складність методу, чутливість до шуму та викидів, коректність вибору кількості сусідів та метрики відстані.

На практиці реалізація даного методу можлива за допомогою класу KNeighborsClassifier() бібліотеки sklearn.neighbors [17].

Також одним із найпоширеніших алгоритмів у машинному навчанні для задач бінарної та багатокласової класифікації, стала **логістична регресія**. Дана модель на основі використання логістичної функції може допомогти дослідити, чи статус платежа буде успішним, чи ні. Логістична регресія може бути також використана для прогнозування ймовірностей або використовуватись як базова модель в ансамблевих методах [18]. До того ж при застосуванні класу LogisticRegression() бібліотеки scikit-learn до процесу автоматично включена регуляризація - техніка, що допомагає контролювати перенавчання моделі та покращує її загальну здатність до узагальнення на нові дані. Це досягається шляхом введення додаткових обмежень (штрафів) до функції втрат (loss function) під час тренування моделі [19].

Перевагами такого підходу можуть бути:

- Інтерпретованість: логістична регресія надає зрозумілі та інтерпретовані результати. Коефіцієнти регресії можна інтерпретувати як вплив кожної змінної на ймовірність віднесення до певного класу.
- Ефективність при роздільних просторах: логістична регресія добре працює у випадку, коли зразки даних добре розділяються лінійно або нелінійно.
- Масштабованість: логістична регресія працює добре з великими наборами даних, і її можна легко масштабувати на великі обсяги даних.

Водночас до недоліків тут можна віднести можливу мультиколінеарність, вразливість до викидів та лінійну роздільність (може погано справлятися з даними, де залежність між змінними не є лінійною).

Наступна модель - модель **випадкового лісу** (Random Forest), що є одним з популярних методів машинного навчання, який використовується для класифікації і базується на понятті ансамблю дерев рішень [20].

Основна ідея моделі випадкового лісу полягає у створенні великої кількості різних дерев рішень та об'єднанні їх результатів для отримання більш точних та надійних прогнозів. Кожне дерево у лісі навчається на випадковій підмножині даних, а також використовує лише деяку підмножину ознак (зазвичай випадковим чином вибирається кілька ознак для кожного дерева). Така

випадковість допомагає зменшити кореляцію між деревами і забезпечує більшу різноманітність моделі [21].

Однією з основних переваг моделі випадкового лісу є її здатність працювати з великими наборами даних, враховуючи при цьому багато ознак. Вона може автоматично обробляти пропущені дані та робити оцінку важливості вхідних факторів. Ще одна перевага полягає в тому, що така модель зазвичай не потребує складного налаштування параметрів і має невелику кількість гіперпараметрів, що полегшує її використання. Крім того, вона добре працює з даними, що мають як категоріальні, так і числові ознаки.

Однак, модель випадкового лісу має кілька обмежень. Вона може мати схильність до перенавчання на деяких завданнях, особливо якщо кількість дерев у лісі дуже велика. Також, порівняно з іншими методами, модель випадкового лісу може бути менш чутливою до взаємозв'язку між ознаками та вимагати високий обчислювальний ресурс, особливо при великій кількості дерев рішень.

Останнім до розгляду але не останнім за результативністю стане **класифікатор на основі легкого градієнтного бустингу** (Light Gradient Boosting Machine Classifier) - модель машинного навчання, що використовує алгоритм градієнтного бустингу на основі дерев рішень для класифікації даних [22].

Основним принципом роботи LGBMClassifier є покрокове покращення моделі шляхом додавання слабких моделей, таких як дерева рішень. Так починаючи з одного дерева рішень, LGBMClassifier намагається знайти оптимальний розділ даних, що найкраще розділяє класи цільової змінної. Далі модель визначає вагу помилок, яка вказує, наскільки важливо виправити помилки класифікації на кожному кроці і додає наступне дерево рішень, спрямовуючись на зменшення ваги помилок і покращення класифікації. Модель використовує градієнтний спуск для налаштування параметрів дерев, таких як глибина, швидкість навчання і розподіл ваг. Загалом, модель стає ансамблем декількох дерев рішень, де кожне дерево вносить свій внесок у класифікацію [23].

До переваг такого методу можна віднести високу швидкодію завдяки використанню оптимізованих структур даних, ефективне використання ресурсів, що допомагає при роботі з великими датасетами без перенавантаження процесора та навіть підтримка роботи з категорійними змінними.

Недоліки тут теж існують, серед яких стандартні ля більшості моделей чутливість до перенавчання, вразливість до шуму та викидів та потреба в коректному налаштуванні параметрів роботи оптимізатора.

У наведених вище моделях класифікації в першу чергу важливо розуміти правильність віднесення клієнта до класів з платежем і без, і вже на основі отриманих оцінок обирати модель для фінального використання в моделюванні.

Так зазвичай слідом за побудованою моделлю класифікації йде побудова confusion matrix – матриці, що співставляє правильно та неправильно класифіковані об’єкти та, в контексті поточної задачі, має наступний вигляд:

Таблиця 2.2

Confusion matrix

	Прогнозовані платники (B)	Прогнозовані неплатники (N)
Реальні платники (Br)	TP	FN
Реальні неплатники (Nr)	FP	TN

Джерело: складено автором на основі [24].

Ще показники вище можна виразити через ймовірності, де $TP = P(BBr)$, $FN = P(NBr),\ FP = P(BNr),\ TN = P(NNr)$.

Надалі отримані показники з матриці вище можна конвертувати у метрики, на основі яких можна приймати рішення про порогову ймовірність для віднесення до позитивного класу об’єкта та оцінювати якість роботи того чи іншого класифікатора. Такими метриками зазвичай є precision (точність) та recall (повнота). Для оцінки точності також прийнято використовувати метрику accuracy, проте вона має місце радше в збалансованих даних, де платник/неплатник розподілені більш в рівних пропорціях.

Якщо говорити про precision – то дана метрика характеризує, скільки платників, що отримали при моделюванні, реально є платниками, і виражається наступною формулою та умовною ймовірністю [24]:

$$precision = \frac{tp}{tp + fp} = \frac{P(BBr)}{P(BBr) + P(BNr)} = \frac{P(Br|B) * P(B)}{P(Br|B) * P(B) + P(Nr|B) * P(B)}$$

$$= \frac{P(Br|B)}{P(Br|B) + P(Nr|B)} = P(Br|B) \quad (2.1)$$

Якщо говорити про recall – то дана метрика характеризує, скільки платників наша модель взагалі здатна ідентифікувати з усіх можливих, і виражається наступним чином [24]:

$$recall = \frac{tp}{tp + fn} = \frac{P(BBr)}{P(BBr) + P(NBr)} = \frac{P(B|Br) * P(Br)}{P(B|Br) * P(Br) + P(N|Br) * P(Br)}$$

$$= \frac{P(B|Br)}{P(B|Br) + P(N|Br)} = P(B|Br) \quad (2.2)$$

Оскільки за своєю суттю показники вище можна вважати конкурентами – тобто чим більше охоплення, тим менше точності, і навпаки – для збалансованості між ними прийнято рахувати в F-measure або f1-score - метрику, яка рахує середнє гармонійне між precision rate та recall rate та обчислюється за формулою нижче [25]:

$$f_1 = 2 * \frac{precision * recall}{precision + recall} \quad (2.3)$$

На основі даного балансування і максимізації f1-метрики, зрештою, і вибиратиметься порогове значення ймовірності для відношення до того чи іншого класу клієнта, а оцінка результативності моделей зводитиметься до простого «чим вище f1-score – тим краще».

Застосування даного підходу до оцінки на практиці можливе за рахунок функції з модуля

Додатково до цього методу можемо розглянути ROC-криву і обраховану за її допомогою площу під кривою False Positive Rate – True Positive Rate – AUC-scrore. Однак, такий підхід краще використовувати до врівноважених даних, що нерелевантно для моделювання платників, яких меншість [25]. Зокрема, така

нерелевантність проявляється при обчисленні False Positive Rate, оскільки в даному показнику втілюється відношення прогнозованої кількості платників, які ними не стануть (FP), до всіх неплатників (FP+TN) – тому фінальне значення метрики AUC може бути завищеним.

2.3 Підходи до визначеності важливості факторів та їх відбору для подальшого моделювання

Процес оцінки важливості того чи іншого фактору спрямований на вибір найбільш інформативних та значущих ознак з набору даних, що впливають на прогнозну здатність моделі.

Основна мета відбору факторів полягає в тому, щоб включити до моделі тільки найбільш важливі ознаки, що дозволяє досягти декількох переваг:

- Зменшення складності моделі: вибірка найбільш важливих ознак дозволяє зменшити кількість зайвих даних, що може поліпшити інтерпретованість, розуміння та ефективність моделі;
- Зниження ризику перенавчання: надмірна кількість незначущих ознак може призвести до перенавчання моделі, коли вона вчиться шуму та непотрібним закономірностям в даних і відбір факторів допомагає зменшити цей ризик, зберігаючи лише релевантні ознаки;
- Покращення швидкодії та ефективності: обмеження кількості ознак може значно оптимізувати обчислювальну складність та час, необхідний для тренування моделі, що особливо важливо при роботі з великими обсягами даних.

Процес відбору факторів може включати різні методи, такі як статистичний аналіз, коефіцієнти важливості, методи зменшення розмірності, алгоритми відбору ознак та багато інших. Вибір конкретного методу залежить від типу даних, задачі прогнозування та вимог щодо моделі.

Першою такою базовою моделлю для відбору незалежних змінних є кореляція за Пірсоном, результатом роботи якої є коефіцієнт, що вимірює зв’язок між ознакою та змінною, що прогнозується [26]. Даний коефіцієнт можна отримати при обчисленні за формулою нижче:

$$R_{xy} = \frac{\sum_{m=1}^{n}(x_m - \bar{x})(y_m - \bar{y})}{\sqrt{\sum_{m=1}^{n}(x_m - \bar{x})^2 \sum_{m=1}^{n}(y_m - \bar{y})^2}} \quad (2.4)$$

де:

x_m *та* y_m *– випадкові змінні;*

$\bar{x}$ та $\bar{y}$ – вибіркові середні;

n – кількість спостережень.

За реалізацію даної формули в Python відповідає функція corrcoef() бібліотеки NumPy, результатом виклику якої є квадратна матриця кореляції, де кожен елемент (i, j) представляє кореляцію між і-тою і j-тою змінними. Значення кореляції знаходяться в діапазоні від -1 до 1, де 1 вказує на повну позитивну кореляцію, -1 вказує на повну негативну кореляцію, а 0 вказує на відсутність кореляції між змінними [27]. При цьому даний підхід не відповідає на питання каузальності – тобто наявність зв'язку ще не говорить про те, що залежна ознака змінюється саме від змін у незалежній ознаці.

Наступним підходом до класифікації ознак є статистичний метод хі-квадрат (chi-square), який використовується для визначення залежності між двома категоріальними змінними. Суть даного методу полягає у спростуванні нульової гіпотези про відсутність залежності шляхом розрахунку статистики хі-квадрат за формулою, наведеною нижче [28]:

$$\chi^2 = \sum_{m=1}^{n} \frac{(O_m - E_m)^2}{E_m} \tag{2.5}$$

де:

O_m – спостереження типу m;

E_m – очікуване значення типу m, відповідно до нульової гіпотези;

n – кількість спостережень.

У контексті відбору ознак при моделюванні, метод chi-square може бути використаний для класифікації факторів за їх важливістю або значущістю відносно цільової змінної на основі методу chi2 класу SelectKBest() в Python [29].

Однак тут, перш ніж переходити до самого процесу класифікації, доречно буде нормалізувати фактори. По-перше, сам метод передбачає на вхід невід'ємні дані за умови рівномірного їх розподілу, по-друге, таким чином можна зменшити вплив розмаху значень, забезпечити більш однорідну шкалу для ознак, урівноважити вплив кожної з них на статистику і так забезпечити стабільність оцінки при незначних змінах у вхідних даних.

Так бажаного результату можна досягнути шляхом використання класу MinMaxScaler(), в основі роботи якого лежить формула (2.6):

$$x_m^* = \frac{x_m - x_{min}}{x_{max} - x_{min}} \quad (2.6)$$

де:

x_m *– поточне значення ознаки в ряді;*

x_{min} *та* x_{max} *– мінімальне та максимальне значення, наявні в ряді ознак.*

Дві попередньо розглянуті оцінки можна віднести до методів-фільтрів, що базуються на статистичних характеристиках даних і використовують певні метрики для оцінки важливості ознак і відбору найбільш інформативних з них. Основна ідея полягає в тому, що ознаки, які мають високу кореляцію з цільовою змінною або великий вплив на модель, мають вищу ймовірність бути корисними у прогнозуванні [30].

Перевагами таких методів можна вважати їх простоту, обчислювальну ефективність та збереження інформації, оскільки не залежать від побудови моделі. Недоліками ж можна вважати їх абстрактність, оскільки вони оцінюють важливість змінних незалежно від конкретної задачі або моделі та можуть не враховувати взаємодію між ознаками. Також можна відмітити їх недостатню універсальність та ігнорування моделі – тобто не використання інформації про саму модель, що може призводити до відбору ознак, які не є оптимальними для конкретної моделі.

Враховуючи переваги та недоліки методів фільтрування, їх застосовують як перший крок у відборі ознак для подальшого використання у моделях машинного навчання. Вони допомагають швидко відсіяти менш інформативні ознаки, зменшуючи розмірність простору ознак та покращуючи швидкість обчислень. Однак, для більш точного та контекстного відбору ознак часто використовуються методи обгортки та вбудовані методи, які враховують модель та її властивості. Тож далі розглянемо ще й такі підходи.

Метод обгортки (wrapper) є одним з підходів до відбору, що базується на ітеративному процесі, в якому модель машинного навчання використовується

для оцінки набору ознак, і відбираються ті ознаки, які мають найбільший вплив на прогнозування моделі. Основна ідея методу обгортки полягає в тому, що модель машинного навчання використовується як "обгортка" навколо вибору ознак [31]. Процес відбору ознак включає наступні кроки:

1. Вибір початкового набору ознак, який може бути зроблений на основі експертних знань, апріорної інформації або інших методів відбору ознак;
2. Використання моделі машинного навчання для оцінки впливу кожної ознаки на прогнозування;
3. Відбір підмножини ознак з найвищим впливом на прогнозування за допомогою певного критерію;
4. Повторення кроків 2-3 з використанням вибраної підмножини ознак досягнення кінцевого критерію відбору ознак (наприклад, досягнення заданої кількості потрібних ознак або точності моделі);
5. Оцінка кінцевої моделі на незалежному наборі даних для перевірки її ефективності.

Метод обгортки є досить обчислювально витратним, оскільки вимагає багаторазового навчання моделі з різними підмножинами ознак. Однак, він може бути ефективним для відбору в складних задачах, де інші методи можуть бути менш ефективними.

Так, третім підходом до вибору найважливіших змінних при моделюванні є метод рекурсивного видалення ознак (Recursive Feature Elimination, RFE), що є одним із представників обгортки при відборі ознак. Метод RFE є ітеративним і використовує модель машинного навчання для оцінки впливу ознак на прогнозування. Він дозволяє відібрати найбільш важливі змінні шляхом поступового видалення менш важливих. В результаті отримується підмножина ознак, яка має найбільший вплив на модель прогнозування [32-33].

Моделлю навчання, на основі якої побудований процес оцінки, стала логістична регресія, про яку вже згадували в попередньому підрозділі.

У процесі рекурсивного видалення ознак логістична регресія використовується для оцінки важливості ознак. Видаляються ознаки з

найнижчими абсолютними значеннями вагових коефіцієнтів – тобто ті, що мають менший вплив на прогнозування і є менш інформативними для моделі.

Але насправді, якщо говорити про оцінку факторів шляхом побудови базової моделі, то нерідко самі моделі мають вбудовані методи відбору, що використовуються для автоматичного визначення найбільш важливих ознак в наборі даних під час процесу навчання моделі. Вбудовані методи використовують властивості моделі безпосередньо, тоді як методи обгорток будують та використовують модель як частину процесу відбору ознак.

До переваг вбудованих методів можна віднести ефективність та простоту використання: ці методи відбору ознак зазвичай працюють швидше, оскільки вони використовують властивості моделі безпосередньо та не потребують додаткового проходу по даних або використання додаткових алгоритмів, їх легко використовувати, оскільки вони доступні як частина бібліотек машинного навчання, таких як scikit-learn.

Звісно є і недоліки у вигляді обмеженої варіативності та залежності від моделі, проте якщо модель все ж підтримує пряме визначення ознак – такий підхід може бути корисним [34].

Так за рахунок використання класу SelectFromModel() бібліотеки scikit-learn можна отримати найважливіші ознаки автоматично [35]. При цьому базовою моделлю для застосування даного підходу було обрано модель випадкового лісу.

При вирішенні поточного завдання з відбором найважливіших ознак доречною є ще одна перевага випадкового лісу, що не озвучувалась раніше, а саме: наявність вбудованої функції оцінки важливості ознак. За допомогою аналізу важливості ознак можна визначити, які з них найбільше впливають на прогнози моделі, що дає змогу зрозуміти, які аспекти даних є найбільш суттєвими для вирішення конкретної задачі та дозволяє здійснювати вибірку та оптимізацію ознак для поліпшення прогнозу.

Тобто наразі отримали 4 підходи до відбору ознак, практичне застосування яких буде реалізовано після етапу підготовки потрібного датасету.

Висновки до розділу 2

Найпростішим та найбільш вживаним підходом до розуміння патернів поведінки клієнтів на продукті та обсягу потенційних майбутніх надходжень від них є когортний аналіз. Однак через ряд недоліків, які йдуть паралельно з його перевагами, варто все ж розглянути інші, більш точніші, підходи, що використовуються в цілях саме моделювання платників та грошей від них. Такими підходами стали методи машинного навчання, а саме 4 класифікаційні моделі k-NN, логістична регресія, випадковий ліс та LGBMClassifier. Кожна з них має свої особливості використання, ряд плюсів та мінусів, однак всі розглянуті в теорії моделі надалі будуть використані на практиці з подальшим співставленням їх результативності. При чому результативність таких моделей теж має певні визначенні підходи вимірювання, як от оцінка точності, повноти та їх збалансованого значення – f1-score, що теж було розглянуто в даному розділі.

Однак перед переходом до безпосереднього моделювання, важливим етапом є процес відбору найбільш інформативних та значущих ознак з набору даних, що впливають на прогнозну здатність моделі. Існують різні підходи до реалізації даного процесу, тут зокрема були розглянуті методи-фільтри, методи-обгортки та вбудовані методи, що надалі будуть комплексно використовуватись при роботі на етапі підготовки даних до моделювання.

РОЗДІЛ 3. МОДЕЛЮВАННЯ ДОХОДУ КЛІЄНТІВ НА ОСНОВІ КЛАСИФІКАЦІЙНОЇ МОДЕЛІ

3.1 Підготовка даних для їх використання при моделюванні платників

Дані, що використовуватимуться при моделюванні, були отримані від української IT-компанії Quarks, продуктами якої є додатки та веб-сайти у сфері social discovery. Проте варто зауважити, що дані були попередньо оброблені та не відображають реальної ситуації, хоча і були використані з метою аналізу та моделювання даних про доходи.

Так на основі побудованого SQL-запиту, який можна проглянути детальніше в додатку А, були зібрані дані, що зазвичай є відомими на етапі залучення клієнтів та купівлі ними підписки:

Таблиця 3.1

Огляд даних про клієнтів

name	**description**	**type**
client_id	ID клієнта	ID
scheduled_dt	дата, на яку заплановане продовження підписки	DATE
root_order_id	ID підписки	ID
gender	стать клієнта	BOOL
lt_user	lifetime клієнта	NUM
os_family	операційна система	CAT
age	вік клієнта	NUM
hours_after_reg_buy	через скільки годин після реєстрації клієнт оформив підписку	NUM
country_code	країна реєстрації	CAT
cac	вартість залучення клієнта	CAT
retry_amount	кількість спроб продовжити підписку	NUM
period	на який період оформлена підписка	NUM
income	дохід від продовження підписки	NUM
currency_id	валюта, в якій оформлена підписка	CAT
percent_off	розмір знижки	NUM
psp	провайдер платежу	CAT
last_response_code	відповідь при останній спробі продовжити підписку	CAT
bank	банк, в якому оформлена картка користувача	CAT
was_3ds	чи був 3D Secure при проведенні платежа	BOOL
card	вид карти	CAT

Продовження таблиці 3.1

name	description	type
card_country_code	країна оформлення картки	CAT
success_orders	кількість попередніх вдалих оплат	NUM
not_success_orders	кількість попередніх невдалих оплат	NUM
insufficients	кількість невдалих оплат клієнта по причині недостатньої кількості коштів на картці	NUM
root_gross	на яку суму користувач вже оформив дану підписку	NUM
not_root_gross	яку суму користувач витратив без врахування даної підписки	NUM
recurrent_status	чи відбулось продовження підписки	BOOL
current_dt	дата поточного платежа	DATE

Джерело: складено автором.

Наступним етапом є імпорт даних в Python, при чому весь код, що використовуватиметься надалі для роботи з моделюванням в даному програмному середовищі буде розміщено в додатку Д.

Так в першу чергу необхідно прибрати поля, які будемо вчитися прогнозувати, а саме статус успішності повторного платежу. Також фактори, які залишились, можна поділити за типами даних в них на 4 логічні групи – num_fs, cat_fs, date_fs, bool_fs.

Також необхідно забезпечити повноту та правильність інтерпретації вхідних даних для аналізу. Під повнотою тут мається на увазі зробити перевірку на наявність пропущених значень з їх подальшим заповненням. У моєму випадку такими полями були phone, bank та card_type. Під правильністю інтерпретації – обробку категорійних даних з групи cat_fs.

При первинному аналізі даних також важливо було зрозуміти, скільки унікальних користувачів є в датасеті і з якою частотою ті роблять покупки. Так 35% клієнтів роблять 2 платежі, 50% - 3-4 платежі. Але під платежами тут маються на увазі як купівля підписки, так і факт витрати грошей на сайті на інші види покупок. Якщо розглянути розподіл саме у розрізі підписок, то 37% підписок продовжуються на другий термін, 26% на 3 і 21% на 4 терміни.

Для подальшого розгляду розподілів значень по клієнтах візьмемо дані з останнього платежу – так кожному клієнту відповідатиме один рядок даних, а не кілька. Основна причина цього рішення – це зменшення ваг окремих клієнтів, які

могли і можуть здійснити більше від інших покупок і негативно вплинути на стабільність моделі, оскільки будуть мати більший вплив на модель, незалежно від інших характеристик. До того ж таким чином зменшується розмір вхідного датасету, що пришвидшить процес подальшої роботи з ним.

Найчастіше в датасеті зустрічаються клієнти з покупкою в 4 лайфтайм – тобто в 4 день після реєстрації. За розподілом показника hours_after_reg_buy видно, як клієнти найчастіше оформлюють підписку відразу після реєстрації і зазвичай першим періодом для старту підписки обирають 3-денний. Тому і виходить картина, коли на 4 день ми вже маємо інформацію про те, чи клієнт продовжив підписку, чи ні. По віковій приналежності більшість клієнтів сконцентрована в діапазоні 30-45 років. Також за розподілом retry_amount можна помітити, що більшість платежів проводиться відразу і лиш 46% потребували повторних спроб списання коштів у період подовження підписки. При огляді періоду можна помітити, що найчастіше це 30 днів продовження підписки, далі йде 15, 8, 31. Такий розподіл по днях пов’язаний з обраним користувачем тарифним планом та кількістю попередньо невдалих платежів. При цьому лише 48% клієнтів отримували знижку у розмірі 50-75%. У більшості з них при цьому був принаймні 1 успішний попередній платіж та 0-3 неуспішних, де серед причин неуспішних можна виділити причину недостатності коштів на картці при проведенні транзакції. Що стосовно доходу, який одержується від купівлі/подовження підписки та інших транзакціях – тут більша частина клієнтів зосереджена на нижчих значеннях доходу і навпаки – менше клієнтів мають більші показники. Детальніше з графіками, на основі яких було сформовано висновки вище можна ознайомитись в додатку Б на рис.Б.1-Б.12.

Після першого етапу з розглядом розподілів кількісних величин можемо перейти до оцінки кореляції між факторами, тобто перевірки гіпотези про наявність мультиколінеарності. Загалом мета даного процесу при побудові регресійної моделі – перевірити лінійну залежність між факторами і якщо така присутня – «вилікувати» її [36]. У моєму випадку така залежність сильно простежується між факторами percent_off та period: тут вона полягає в тому, що

чим більша знижка, тим менший період обирається користувачем для підписки. Тому далі краще буде залишити у факторах лише одну змінну задля більш точного результату роботи моделі – до прикладу прибрати величину знижки і лишити лише період. Відображення кореляції між ознаками можна детальніше проглянути у додатку Б на рис. Б.13.

Далі за допомогою матриці побудованих графіків розсіювання – pairplot – можна розглянути зв'язки між різними наборами чисельних змінних в num_fs. Зокрема цікавим видався зв'язок між величиною первинного платежу по підписці та тим, скільки клієнт загалом платить: чим більша плата за підписку з самого початку – тим більше надалі виручки приноситься за весь період життя на продукті. Графіки розсіювання наведені в додатку Б на рис.Б.14-Б.15.

Також у додатку В аналогічно наведені розподіли по інших типах змінних. Якщо розглядати категоріальні, то їм притаманна ситуація, коли здебільшого клієнти приходять зі схожих браузерів одного типу, з однієї країни, за однією ціною закупки, з однією основною валютою оплати тощо. Тобто в кожному розрізі тут є той чи інший лідер, який збирає навколо себе найбільше клієнтів. Що стосовно бінарних змінних – з графіків можна побачити, що найбільше в датасеті чоловіків, оплати яких здебільшого проходили без 3дс. Якщо розглядати змінні, що відповідають датам – дані лежать в межах лютого-квітня 2023. До того ж на графіку із датами запланованих платежів можна простежити періодичність – її теж можна виокремити як фактор, що впливатиме на прогноз, де найбільше значень припадає на п'ятницю.

Надалі підготовлені дані можна розбити на тренувальну та тестову вибірку. Тренувальна вибірка – це підмножина даних, яку використовують для навчання моделі. Модель будується на цих даних, шукаючи закономірності та залежності між різними ознаками. Тестова вибірка – це підмножина даних, яку використовують для оцінки точності моделі. Тестові дані ніколи не беруться до уваги при тренуванні моделі. Зазвичай, дані розбивають у відношенні 70-80% на тренувальну вибірку та 20-30% на тестову вибірку. Залежно від кількості даних та конкретної задачі можна використовувати інші співвідношення розбиття

даних, проте в даній роботі було використано принцип Парето: 80 на 20. Після попередньої перевірки на вдало проведене розбиття на вибірки можна переходити до наступного етапу – стандартизації даних.

Стандартизація є важливим етапом попередньої обробки даних у багатьох методах машинного навчання. Стандартизація означає перетворення ознак таким чином, щоб вони мали середнє значення 0 та стандартне відхилення 1. Це дозволяє зберігати важливу інформацію про розподіл значень ознак, але зменшує вплив значень, які знаходяться віддалено від середнього, що покращує результати багатьох алгоритмів машинного навчання.

Для стандартизації чисельних ознак був використаний клас StandardScaler з бібліотеки sklearn.preprocessing, що за допомогою методу fit_transform() враховує середнє значення та стандартне відхилення з тренувальної вибірки для подальшої стандартизації як тренувальної, так і тестової вибірок [37]. Бінарні ознаки необхідно привести до діапазону значень 0/1 лиш у випадку статі клієнта, яка в датасеті поки має значення m/f.

Наступним етапом буде кодування категоріальних ознак спершу через клас OneHotEncoder, а також через клас CatBoostEncoder. Клас OneHotEncoder, що належить бібліотеці scikit-learn дозволяє виконати кодування категоріальних ознак у вектори з нулями та одиницями [38]. Внаслідок застосування даного підходу створюється матриця виду «один з К» або «не К», що надалі втілюється як додаткові стовпці в датафреймі. В моєму випадку із 23 стовпців в тренувальній та тестовій вибірках утворилося 85 стовпці закодованих категоріальних змінних. Також другим підходом до цього процесу, який згадувався раніше, є CatBoostEncoder – метод кодування категоріальних ознак за допомогою алгоритму CatBoost. Даний алгоритм використовує градієнтний бустінг для ефективного кодування категоріальних ознак з врахуванням цільової змінної [39]. На відміну від попереднього методу тут не створюється матриця та не додаються нові стовпці з даними – категорії тут переводяться в числовий формат з урахуванням їх частоти та змінної, яка прогнозується.

Далі стандартизовані та закодовані змінні можна об’єднати у тренувальну та тестову вибірки, на основі чого варто переходити до наступного етапу роботи з моделлю – процесу відбору факторів, що найбільше впливають на отримання виручки з клієнтів.

Корисним методом при рішенні даної задачі може бути вибірка за голосуванням (voting selector), суть якого полягає в тому, що кілька базових моделей використовуються для оцінки важливості ознак, і потім важливість кожної ознаки визначається шляхом голосування. Тобто кожна модель «висуває» свій перелік найбільш важливих факторів, і до фінального набору входять ті, які увійшли в більшість таких переліків [40]. Оскільки величина списку, який міститиме найважливіші ознаки, здебільшого задається перед початком роботи моделі, надалі будемо відбирати 90% від наявних даних про клієнтів до первинних списків, що братимуть участь у наступному голосуванні.

Основні результати роботи 4 методів відбору, які розглядалися раніше в другому розділі, розміщені табл. Г1-Г4 додатку Г. На основі них сформувались первинні списки з відібраних ознак за різними підходами, тож тепер можемо переходити до наступного етапу – безпосередньо голосування та формування рейтингу найвпливовіших факторів.

Так максимально можливу оцінку 4 отримали 17 факторів, які збережемо у змінну top1_features: ‘scheduled_dt_dayofweek’, ‘root_gross’, ‘retry_amount’, ‘psp’, ‘period’, ‘not_success_orders’, ‘not_root_gross’, ‘last_response_code’, ‘insufficients’, ‘income’, ‘hours_after_reg_buy’, ‘country_code’, ‘card_country_code’, ‘card’, ‘cac’, ‘bank’, ‘age’.

Ще 2 змінні отримали оцінку 3: ‘lt_user’, ‘currency_id’. Їх теж додамо до іншого, більшого списку top2_features.

З результатами голосування можна ознайомитись в табл. Г.5 додатку Г.
Вже під час самого моделювання можна буде порівняти отримані оцінки з використанням різних наборів незалежних змінних.

3.2. Оцінка ефективності класифікаційних моделей для прогнозування платників та їх порівняння

Якщо переходити до практичного застосування розглянутих раніше моделей класифікації – можемо детальніше ознайомитися з найважливішими параметрами на вході та результатами на виході. При тому для навчання моделей можна подавати ознаки, закодовані різними методами (ohe/cat), та в різній комплектації (top1/top2), що визначалося в попередньому підрозділі. Завдання далі буде обрати такі комбінації вхідних даних, що максимізують цільові метрики і вже за їх використання порівнювати між собою моделі для обрання лідера. На виході при цьому будуть оцінюватися значення roc-auc (площа під ROC-кривою), precision, recall, f1-score, y-perc (середнє значення реальних платників в наборі даних), y_hat_perc (середнє значення прогнозованих платників наборі даних).

Так під час використання методу ***k-NN*** найважливіші параметри, які варто подати на вхід для KneighborsClassifier() є кількість сусідів та метод обчислення відстані між ними. За замовчуванням береться k=5 та Евклідова відстань – що доволі непогано для вирішення поточної задачі, оскільки непарна кількість сусідів дозволяє уникнути проблем при голосуванні та визначенні більшості, а рекомендована їх кількість лежить в діапазоні 5-10.

У результаті роботи моделі були отримані наступні оцінки залежно від поданих на вхід ознак:

Таблиця 3.2

Оцінка моделі k-NN

feature	*split*	*thresh*	*roc_auc*	*precision*	*recall*	*f1_score*	*y_perc*	*y_hat_perc*
ohe	Train	0.4	0.778808	0.762652	0.628837	0.689310	0.266822	0.220005
	Test	0.4	0.718097	0.628432	0.571280	0.598495	0.285676	0.259695
top1	Train	0.4	0.770936	0.757675	0.613249	0.677855	0.266822	0.215961
	Test	0.4	0.717265	0.637552	0.562394	0.597619	0.285676	0.251998
top2	Train	0.4	0.773642	0.764257	0.616488	0.682466	0.266822	0.215232
	Test	0.4	0.718815	0.641321	0.563717	0.600020	0.285676	0.251107

Джерело: розрахунки автора.

Якщо порівнювати результати, які отримали на тестових наборах даних при фінальному прийнятті рішення, то в даному випадку найкращої оцінки за f1-score було досягнуто за використання ознак, що були закодовані методом catboost і при відборі методом voting отримали оцінки 3-4.

Далі при використанні ***логістичної регресії*** для моделювання платників був замінений звичайний solver = 'lbfgs' на 'liblinear', що включає в себе оптимальний процес роботи з датасетом, як в даній роботі, та можливість застосування двох видів регуляризації. В даному випадку залишили l2 регуляризацію, як за замовчуванням. Вона використовує квадратичну норму ваг (L2 норму) для додаткової складової в функції втрат, що призводить до штрафу за великі значення ваг моделі.

За результатами роботи LogisticRegression() отримали наступні оцінки:

Таблиця 3.3

Оцінка моделі логістичної регресії

feature	***split***	***thresh***	***roc_auc***	***precision***	***recall***	***f1_score***	***y_perc***	***y_hat_perc***
ohe	Train	0.304	0.742870	0.565122	0.674688	0.615063	0.266822	0.318554
	Test	0.304	0.744689	0.560544	0.712895	0.627606	0.285676	0.363320
top1	Train	0.389	0.731217	0.607688	0.604443	0.606061	0.266822	0.265397
	Test	0.389	0.733878	0.613118	0.625638	0.619315	0.285676	0.291509
top2	Train	0.381	0.731436	0.602781	0.608897	0.605823	0.266822	0.269529
	Test	0.381	0.733292	0.607616	0.629041	0.618143	0.285676	0.295749

Джерело: розрахунки автора.

В цій моделі кращі результати були отримані при кодуванні методом OneHotEncoder() – з утворенням нових стовпців та бінарним представленням кожної характеристики. До того ж метрика f1-score тут піднялась до показника 0,62, що краще на 2 у.о. в порівнянні з попереднім підходом.

Наступним методом до опрацювання на практиці став метод **випадкового лісу.** Якщо використовувати тут усі стандартні значення – є ризик перенавчити тренувальну модель та отримати низьку оцінку при застосуванні навченого алгоритму на тестовій вибірці. Так оптимальним для використання є параметр

max_depth in (10, 15), що відповідає за максимальну глибину дерев, тобто кількість розгалужень в них.

В даному підході отримали наступні значення оцінок:

Таблиця 3.4

Оцінка моделі випадкового лісу

feature	***split***	***thresh***	***roc_auc***	***precision***	***recall***	***f1_score***	***y_perc***	***y_hat_perc***
ohe	Train	0.345	0.769963	0.607741	0.705686	0.653061	0.266822	0.309824
	Test	0.345	0.760455	0.595188	0.715542	0.649839	0.285676	0.343443
top1	Train	0.347	0.779962	0.636126	0.707128	0.669750	0.266822	0.296604
	Test	0.347	0.763971	0.608896	0.710437	0.655759	0.285676	0.333315
top2	Train	0.36	0.779335	0.647098	0.697007	0.671125	0.266822	0.287401
	Test	0.36	0.763781	0.613985	0.704765	0.656250	0.285676	0.327914

Джерело: розрахунки автора.

Аналогічно до результатів в k-NN найбільша оцінка була отримана при використанні списку з 19/23 з оцінками 3-4 при voting-методі відбору ознак з catboost-методом кодування змінних. При тому знову бачимо нового лідера серед розглянутих вище підходів до моделювання за f1-score зі значенням 0,65.

Якщо залишити цей же підхід і додати трохи градієнтного бустингу – отримаємо застосування ***Light Gradient Boosting Machine Classifier*** на практиці.

Таблиця 3.5

Оцінка моделі LGBMClassifier

feature	***split***	***thresh***	***roc_auc***	***precision***	***recall***	***f1_score***	***y_perc***	***y_hat_perc***
ohe	Train	0.318	0.775412	0.634273	0.697108	0.664207	0.266822	0.293255
	Test	0.318	0.767790	0.609144	0.720458	0.660141	0.285676	0.337879
top1	Train	0.324	0.776149	0.635340	0.698120	0.665252	0.266822	0.293187
	Test	0.324	0.767799	0.613107	0.716393	0.660738	0.285676	0.333801
top2	Train	0.319	0.777100	0.635658	0.700271	0.666402	0.266822	0.293944
	Test	0.319	0.767459	0.613327	0.715258	0.660382	0.285676	0.333153

Джерело: розрахунки автора.

Даний алгоритм – це, по суті, покращена версія випадкового лісу, де враховуються попередні помилки для подальшої кращої роботи прогнозу, до того ж весь цей процес класифікації займає значно менше часу, ніж попередні.

Модель LGBMClassifier() містить доволі багато гіперпараметрів, але для старту використаємо значення за замовчуванням.

Так вперше отримали ситуацію, коли найкраща оцінка в моделі була отримана за рахунок використання сету з топовими ознаками, що отримали максимальну оцінку 4 на попередньому етапі відбору – таких було 17/23 факторів. До того ж, з усіх представлених раніше моделей саме ця отримала максимальне значення f1-score=0,66 при 61% точності та 72% охоплення в тестовій вибірці, 77% roc-auc та різницю у розмірі 0,04 при підрахунку середніх значень платників при прогнозуванні та в реальності. Тож надалі саме цей підхід будемо застосовувати для класифікації клієнтів та моделювання доходу.

3.3 Фіналізація процесу класифікації та моделювання згенерованого доходу клієнтами

За дослідженнями з попереднього підрозділу лідером моделей класифікації став підхід на основі LGBMClassifier. Попередньо до уваги бралися стандартні налаштування моделі в програмному середовищі, але насправді можна спробувати покращити оцінки шляхом підбору кращих параметрів. Зазвичай для виконання задачі такого роду використовуються окремі інструменти, як приклад GridSearchCV та RandomizedSearchCV.

GridSearchCV є методом для пошуку найкращих гіперпараметрів моделі шляхом перебору всіх можливих комбінацій значень гіперпараметрів заздалегідь визначеного простору. Використання GridSearchCV дозволяє автоматизувати процес пошуку оптимальних гіперпараметрів моделі, що зменшує необхідність у ручному переборі та налагодженні [41].

Основна перевага GridSearchCV полягає в тому, що він дозволяє систематично перебрати всі можливі комбінації гіперпараметрів та знайти найкращі значення для вказаної метрики. Це допомагає покращити якість моделі та знизити ризик перенавчання.

Однак, необхідно враховувати, що GridSearchCV може бути витратним за часом, особливо якщо простір гіперпараметрів великий або датасет великий. У таких випадках можна розглянути використання більш ефективних алгоритмів оптимізації гіперпараметрів, наприклад таких як RandomizedSearchCV. Останній в свою чергу реалізовує підбір параметрів моделі шляхом випадкового вибору значень заздалегідь визначеного простору гіперпараметрів. Він, на відміну від GridSearchCV, дозволяє зменшити обчислювальні витрати шляхом випадкового вибору підмножини комбінацій гіперпараметрів для оцінки [42].

Тож, зважаючи на явні переваги випадкового пошуку параметрів моделі, спробуємо провести оптимізацію її найважливіших параметрів, а саме:

- ***n_estimators*** – визначає кількість дерев, які будуть побудовані у градієнтному бустингу. Кожне дерево будується з метою виправлення помилок попереднього дерева. Зазвичай більша кількість дерев забезпечує

більшу потужність моделі, але може призвести до перенавчання, тому важливо знаходити оптимальне значення для конкретної задачі;

- ***learning_rate*** – визначає швидкість навчання моделі, тобто керує внеском кожного дерева у композицію. Менші значення learning_rate збільшують загальну кількість дерев, що потрібно для досягнення тієї ж потужності моделі, але збільшують час навчання. Вибір оптимального learning_rate потребує балансу між швидкістю навчання та точністю моделі;
- ***max_depth*** – визначає максимальну глибину кожного дерева. Збільшення max_depth дозволяє моделі розраховувати більш складні взаємозв'язки в даних, але може призвести до перенавчання;
- ***min_child_samples*** – визначає мінімальну кількість прикладів, що потрібно для створення нового поділу у внутрішньому вузлі дерева. Він контролює регуляризацію моделі шляхом обмеження росту дерев. Більші значення min_child_samples зменшують схильність до перенавчання, але можуть призвести до втрати інформації в деревах.

Значення параметрів вище, що задаються для роботи алгоритму, лежатимуть в наступних діапазонах:

- 'n_estimators': np.arange(0, 2000, 50),
- 'learning_rate': np.arange(0.005, 0.5, 0.005),
- 'max_depth': np.arange(5, 300, 3),
- 'min_child_samples': np.arange(10, 500, 10)

Після запуску пошуку найкращі параметри були визначені як: {'n_estimators': 1750, 'min_child_samples': 360, 'max_depth': 161, 'learning_rate': 0.005}. Оцінки фінальної моделі з таким підходом виглядають наступним чином:

Таблиця 3.6

Оцінка оптимізованої моделі LGBMClassifier

feature	***split***	***thresh***	***roc_auc***	***precision***	***recall***	***f1_score***	***y_perc***	***y_hat_perc***
top1	Train	0.342	0.772527	0.644122	0.682229	0.662628	0.266822	0.282608
	Test	0.342	0.768158	0.622651	0.707884	0.662538	0.285676	0.324781

Джерело: розрахунки автора.

Так отримали незначне але все ж покращення показника f1-score з 0,661 до 0,663, та підвищила точність з 61% до 62% не без зниження охоплення з 72% до 71%.

Тепер перейдемо до детальнішого розгляду отриманих показників під час класифікації. Наглядно видно конкуренцію між охопленням та точністю залежно від порогового значення ймовірності, від якої об'єкт буде класифіковано як позитивний на наведеному графіку:

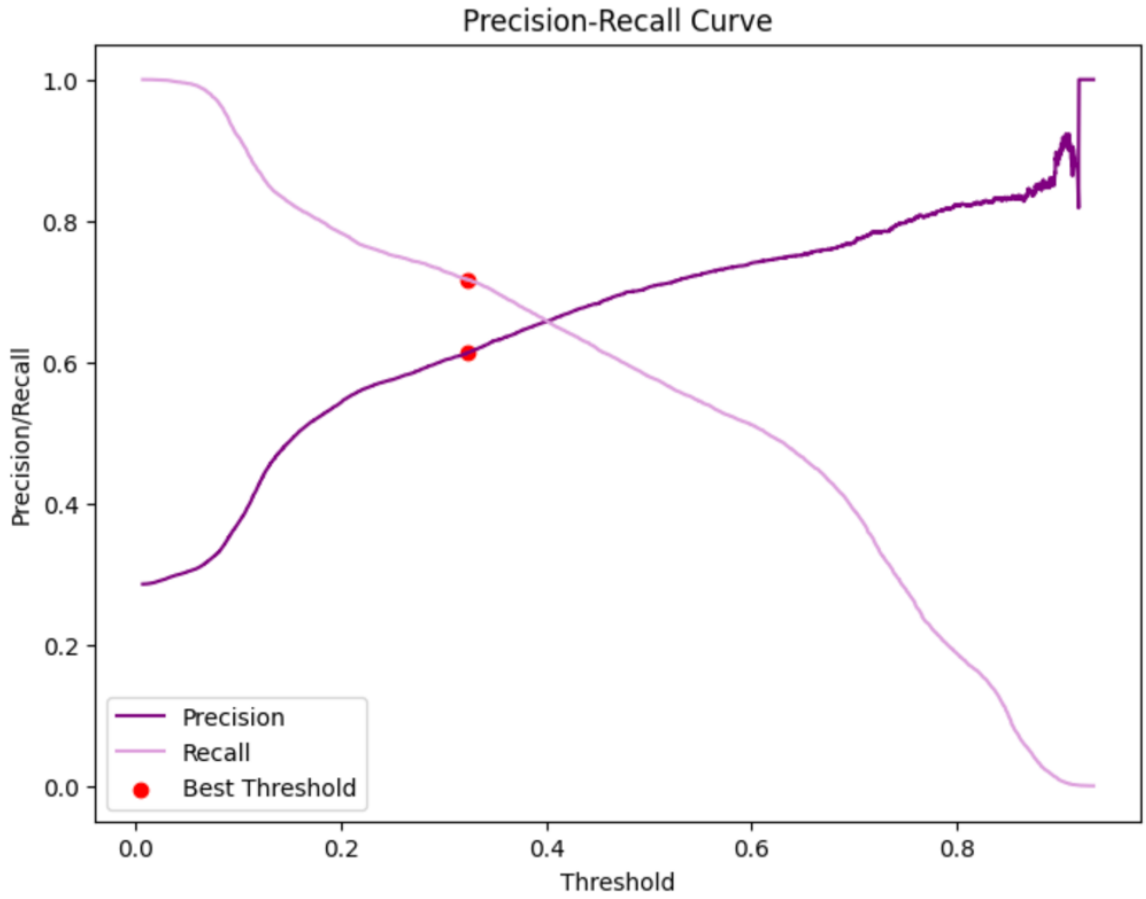


Рис.3.1. Залежність значень precision та recall від порогової ймовірності (threshold)

Джерело: розрахунки автора.

Також поєднання даних показників можна візуалізувати дещо інакше, як на графіку нижче. Тут чим вищим і опуклішим є графік – тим краща якість побудованої моделі оскільки ми менш схильні жертвувати точністю при більшому значенні повноти і навпаки.

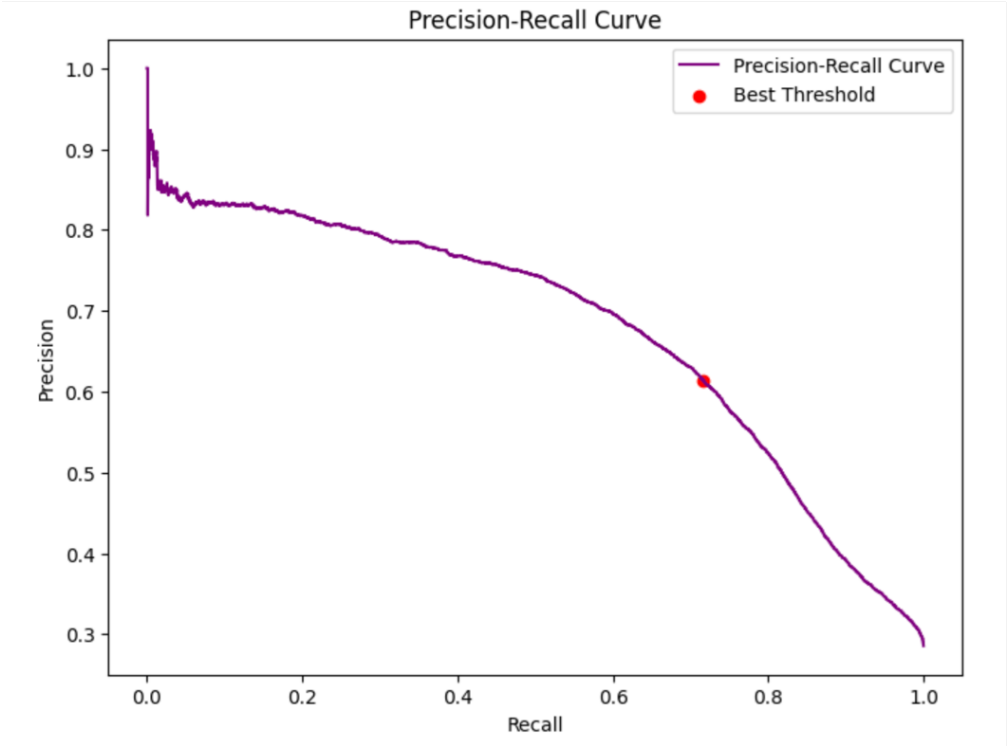


Рис.3.1. Взаємозалежність значень precision та recall

Джерело: розрахунки автора.

Також що стосовно ROC-кривої з відображенням на її основі співвідношення True Positive Rate та False Positive Rate отримали наступну візуалізацію:

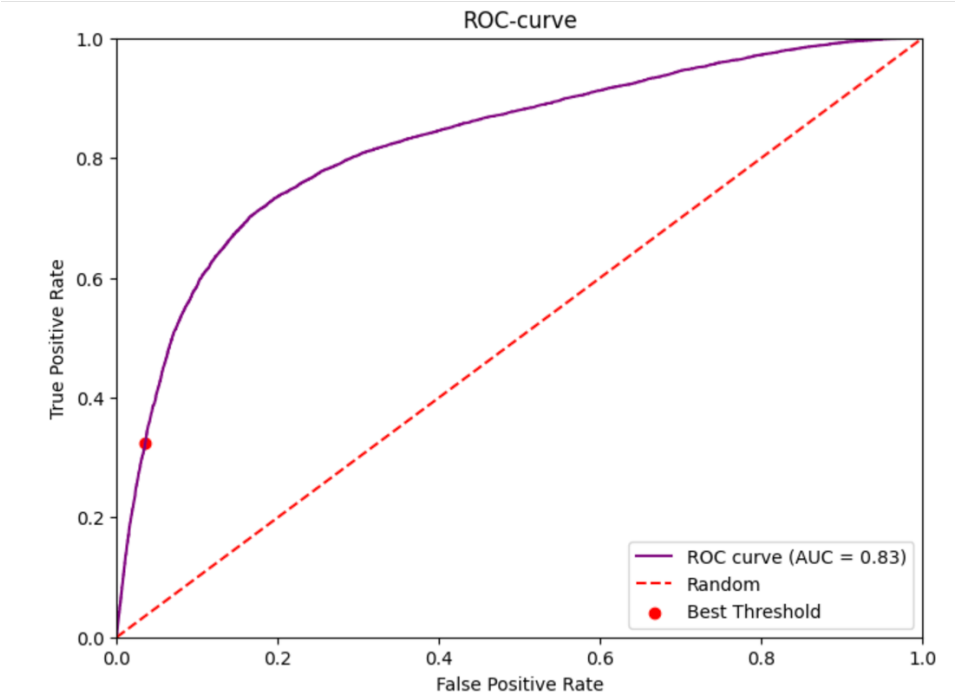


Рис.3.2. Відображення ROC-кривої з обрахунком площі для оцінки моделі

Джерело: розрахунки автора.

Тут отримане значення AUC на рівні 0,83 вказує на те, що модель має добру прогнозну здатність та здатна розрізняти класи з високою точністю, що в принципі видно і з опуклості самої кривої.

Якщо говорити про помилки, то модель неправильно визначила 4 539 клієнтів (або 12%) як потенційних платників – тут бачимо помилку першого роду. Помилка другого роду проявляється в тому, що модель неправильно виділила 3 090 клієнтів (або 8%) такими, що не стануть платниками, хоча в реальності вони таки заплатять. Таке відсоткове співвідношення помилок зокрема знаходить своє відображення у метриках точності та охоплення, про які говорили при оцінюванні моделей класифікації та на які дивитимемось детальніше далі.

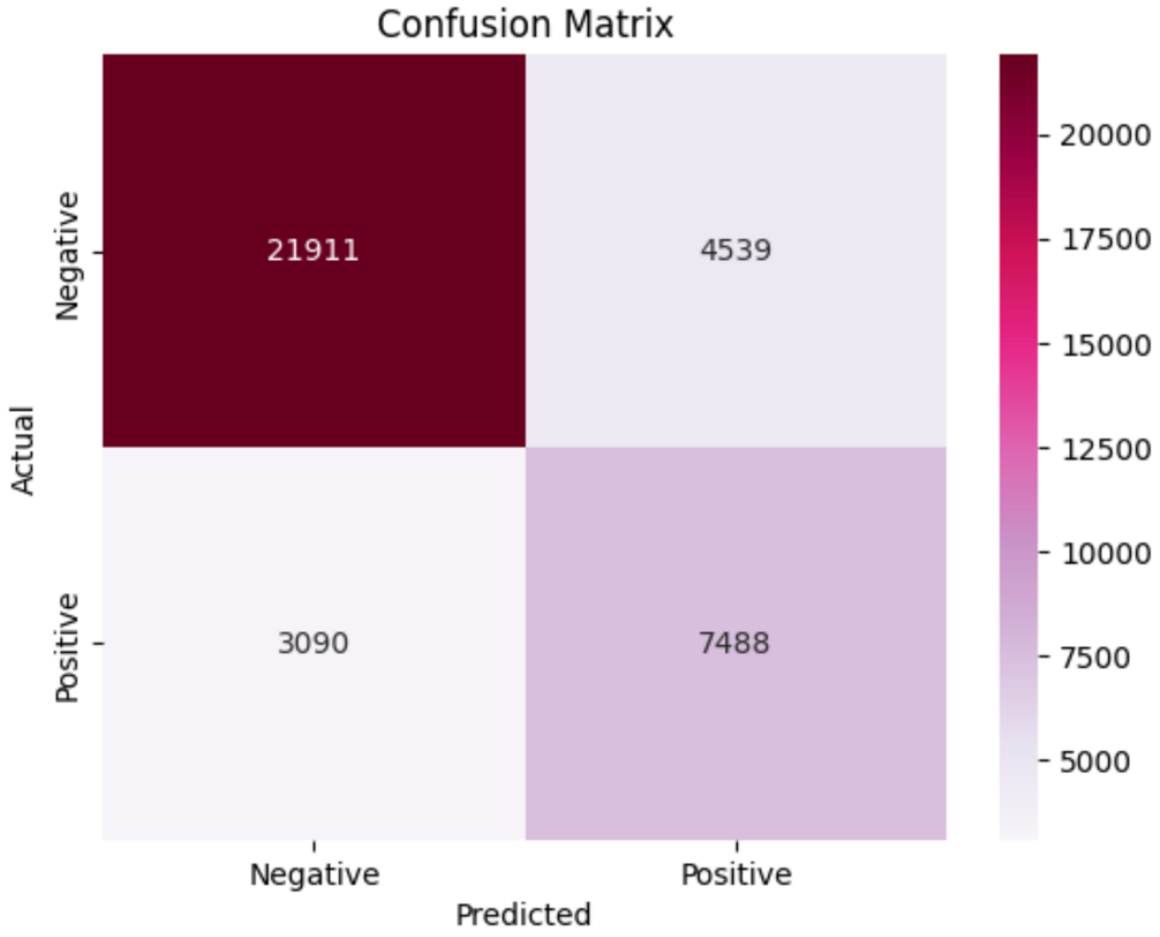


Рис.3.3. Відображення confusion matrix на основі реалізованої моделі класифікації

Джерело: розрахунки автора.

Якщо говорити про важливість факторів, що в даній моделі впливають на прогноз, то маємо наступне їх відображення:

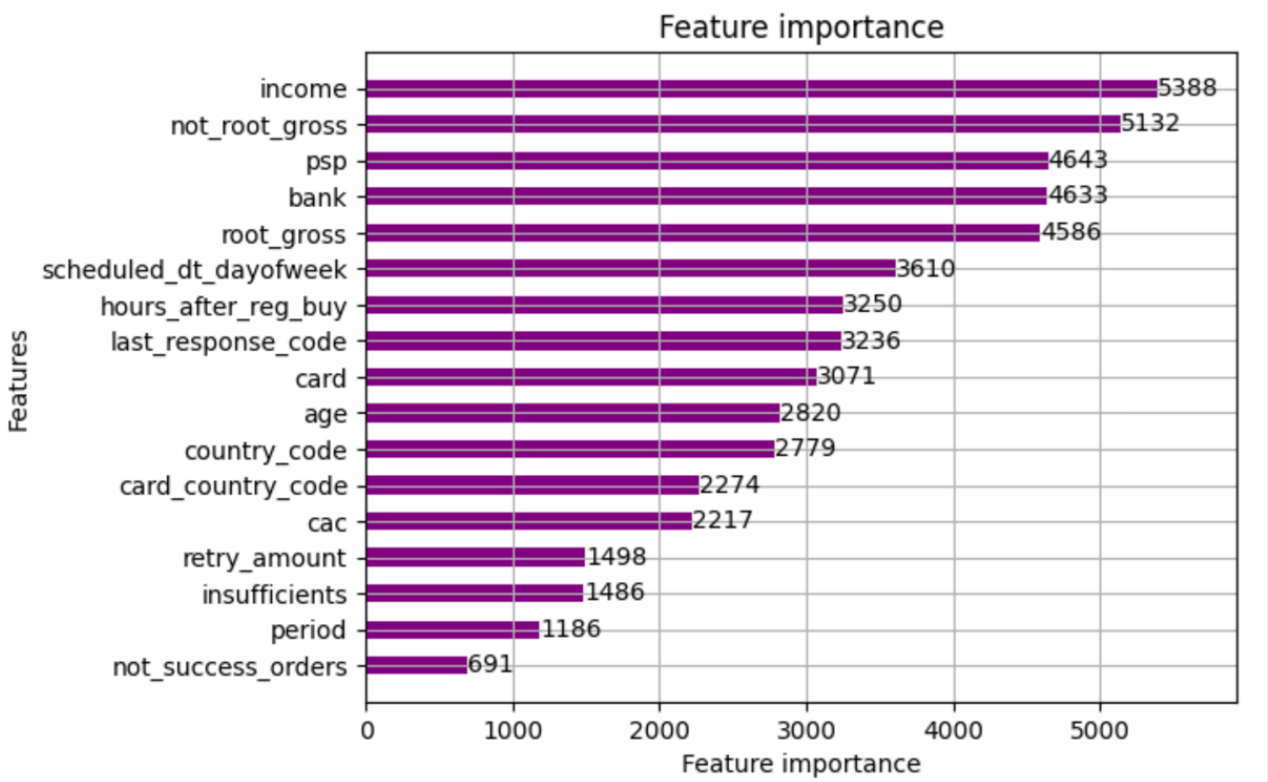


Рис.3.4. Важливість ознак датасету при моделюванні даних про платників
Джерело: розрахунки автора.

Якщо підсумувати, то найбільше впливає саме первинна інформація про платіж клієнта, на скільки швидко він настав, в якому обсязі та через який банк. Також цікаво, що 6 в списку після платіжних даних йде день тижня, в якиий заплановано наступний платіж – нагадаю, що цю ознаку раніше було додано «штучно» після первинного ознайомлення з даними та виявлення в них циклічності, і зараз від цього в досить великій мірі залежить клас клієнта.

Далі описані вище результати глибоких досліджень варто звести до простої відповіді на питання скільки платників буде і скільки грошей ті принесуть?

Що стосовно платників – в тестовій вибірці ми помилились на 14% при віднесенні користувачів до платників, фінальне відображення співставлення в абсолютних величинах можна оглянути на візуалізації нижче:

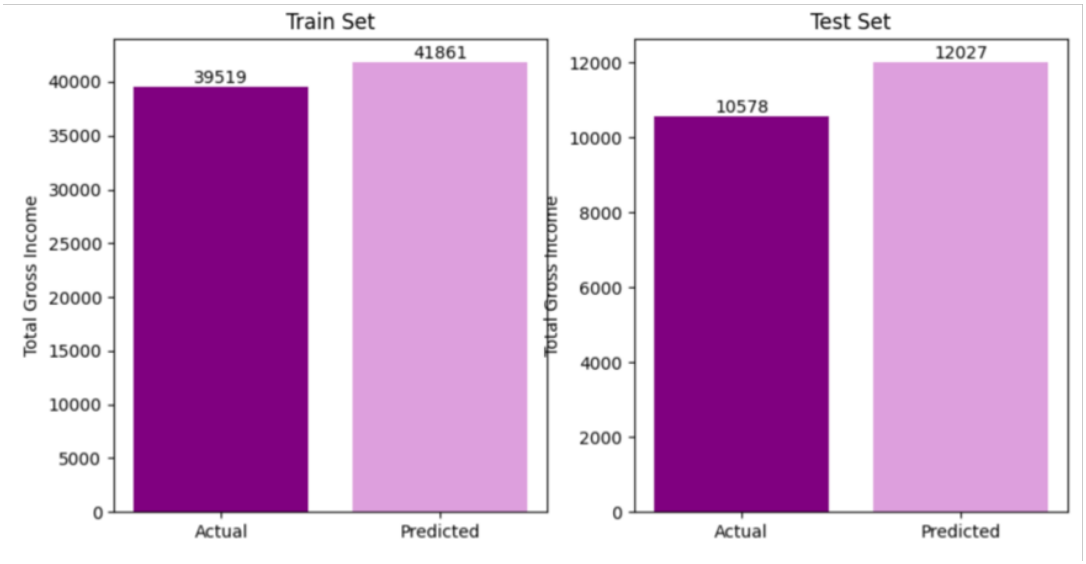


Рис.3.5. Порівняння змодельованих даних про платників з реальними.

Джерело: розрахунки автора.

Що стосується етапу з переведенням результатів у гроші – тут достатньо отриманим при класифікації клієнтам нарахувати середню вартість підписки, відому з первинних платіжних даних. Графік з абсолютними величинами представлений нижче:

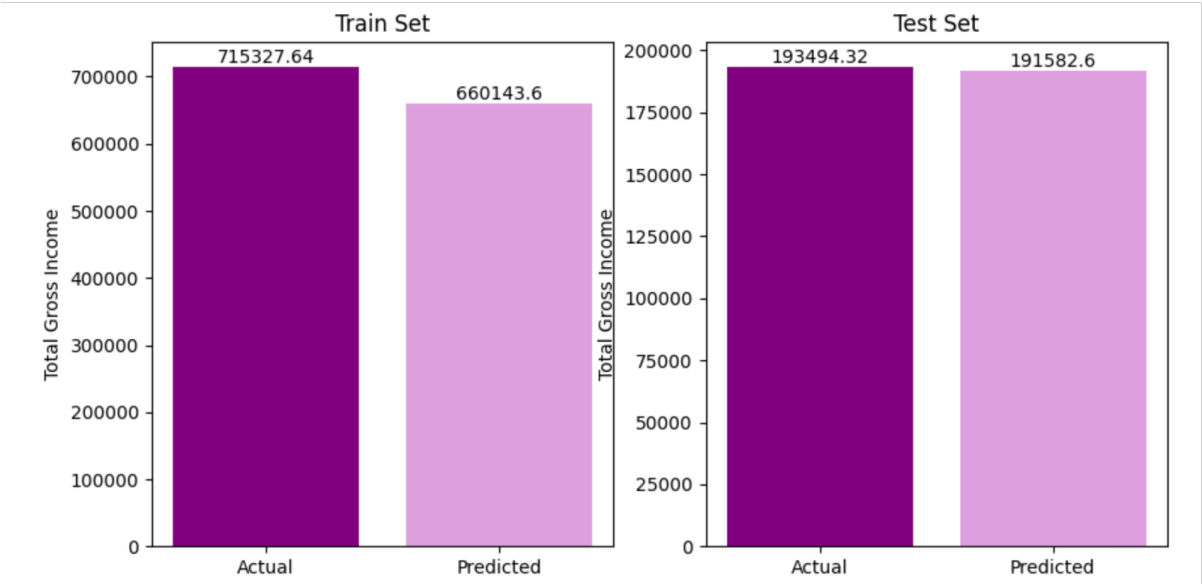


Рис.3.6. Порівняння змодельованих даних про доходи клієнтів з реально отриманими.

Джерело: розрахунки автора.

В результаті маємо 1% відхилення змодельованої виручки від реальної в тестовій вибірці, що є доволі непоганим результатом, на основі якого надалі прийматимуться рішення по підвищенню ефективності маркетингових кампаній.

Висновки до розділу 3

Після формування датасету для подальшого моделювання доходу на основі нього важливим етапом є первинна обробка та підготовка даних до використання в моделях. Зокрема тут мається на увазі обробка пропусків, огляд розподілів вхідних ознак, розділення вибірки на тренувальну та тестову та формування списку з найважливіших ознак, про що згадувалось в минулому розділі а в даному реалізовувалось методом голосування.

Далі, внаслідок почергової побудови класифікаційних моделей на основі різних підходів, були отримані спільні для всіх оцінки результативності і обрано найкращий з них метод за значенням f1-score – LGBMClassifier – метод градієнтного бустингу дерев рішень. Саме він дозволив максимізувати значення точності та охоплення у вигляді збалансованих 66% F-measure, в той час коли інші моделі мали це значення на рівні 60%-65%.

Після додатково проведеної оптимізації гіперпараметрів, що подаються на вхід моделі, отримали значення precision на рівні 62% та recall на рівні 71%. При цьому максимізуючою значення f1-score стала порогова ймовірність у розмірі 0,342 – тобто по цьому значенню надалі і приймалось рішення про приналежність до того чи іншого класу об’єктом. Як наслідок отримали 12% неправильно виділених потенційних платників та 8% упущених клієнтів, що все ж таки стали платниками. Проте якщо все ж таки переходити до фінальної оцінки змодельованого доходу на основі всіх розглянутих вище підходів, отримали різницю у розмірі 1% між реальними та змодельованими даними в тестовій вибірці.

ВИСНОВКИ

Питання ефективного інвестування коштів при залученні нових клієнтів в IT-продуктах призвело до постійної оцінки та вдосконалення підходів в аналізі рекламної діяльності. Так, оскільки останніми роками простежується явне лідерство рекламування саме через інтернет-платформи, великого розповсюдження набули відповідні підходи в маркетингу. В основному серед таких виділяється 3 основні напрями - performance-, product- та brand-marketing. Однак найбільше переваг щодо вимірюваності результативності має в собі саме performance-marketing, в завдання якого входить створення успішної стратегії маркетингових інвестицій задля отримання бажаного рівня прибутку та моніторинг виконання цілей по окупності.

Коли говориться про цілі по окупності – зазвичай мається на увазі зведена unit-економіка – індикатор того, на скільки ефективно інвестується 1$ та чи заробляємо ми достатньо з 1 клієнта або 1 покупки. До того ж в процесі цілепокладання всі розрахунки здійснюються на основі умови про невід'ємний показник %ROI (рентабельність інвестицій), який має декілька методів обрахунку в маркетингу, де той чи інший його вид потрібен на різних етапах оцінки ефективності. Проте і у випадку розрахунку виконання поставлених цілей перед маркетингом у вигляді певного %ROMI (рентабельність маркетингових інвестицій) на n-день, і у випадку підрахунку коефіцієнту співвідношення LTV до CAC (надходження від клієнта за весь час користування продуктом) / (вартість залучення клієнта) при знаходженні unit-економіки – всюди потрібне розуміння того, скільки грошей клієнти, залучені зараз, згенерують колись у майбутньому.

До того ж швидкий розвиток IT-індустрії та постійні зміни на ринку вимагають від компаній постійно покращувати свої продукти. Один з ефективних способів, що забезпечує це постійне покращення - це A/B тестування. В контексті маркетингу під час проведення тестування різного роду гіпотез команди орієнтуються не лише на здешевлення вартості приведення клієнта на продукт, а й на збільшення фінансових показників. І тут часто для

коректної оцінки може бракувати цілісної картини про доходи, що, як і в попередньому пункті з цілепокладанням, може вирішуватись шляхом моделювання грошей, які компанія заробить у майбутньому. Так виникає потреба в розгляді питання моделювання доходів клієнтів.

При дослідженні наявних підходів до отримання прогнозу по виручці своєю простотою та легкістю імплементації відзначився когортний аналіз. Так за рахунок чисельного відслідковування тенденцій залучених раніше користувачів, а саме формування виручки ними впродовж кожного нового дня користування продуктом, можна змоделювати аналогічні дані і для нових когорт з використанням відповідних коефіцієнтів. Проте даний підхід має ряд своїх мінусів, серед яких потреба в детальному розбитті даних, яких має бути вдосталь для того та проблеми з інтерпретацією результатів, а саме з розумінням факторів, що мають суттєвий вплив на поведінку клієнтів, та тим, та хто саме з клієнтів платить і чому.

Саме тому далі виникла потреба у застосуванні інших методів, а саме – методів машинного навчання, де на основі історичних даних модель навчається, та в результаті моделює майбутні дані. Такими методами стали методи 4 класифікаційні моделі k-NN, логістична регресія, випадковий ліс та LGBMClassifier. Кожна з них має свої особливості використання, ряд плюсів та мінусів, однак всі розглянуті в теорії були застосовані на практиці та дали той чи інший результат при роз’язуванні поставленого завдання.

Однак перед переходом до безпосереднього моделювання, важливим етапом став процес відбору найбільш інформативних та значущих ознак з набору даних, що впливають на прогнозну здатність моделі. Так були розглянуті методи-фільтри, методи-обгортки та вбудовані методи, що на етапі підготовки даних до моделювання були комплексно використані за voting-підходом.

Дані, на основі яких відбувалося моделювання, були отримані на основі діяльності ІТ-компанії за попередньої їх шифрування та обробки. Якщо говорити про друге – тут мається на увазі обробка пропусків, огляд розподілів вхідних

ознак, розділення вибірки на тренувальну та тестову та формування списку з найважливіших ознак, про що згадувалось раніше.

Далі було здійснено почергову побудову класифікаційних моделей з оцінкою кожної за основними підходами, розглянутими в методології оцінювання раніше, серед яких зокрема precision (точність), recall (повнота) та f1-score – метрика, що в собі поєднує 2 попередні показники. Так найкращою з розглянутих моделей за значенням f1-score стала модель на основі LGBMClassifier – методу градієнтного бустингу дерев рішень. Саме він дозволив максимізувати значення точності та охоплення у вигляді збалансованих 66% F-measure, в той час коли інші моделі мали це значення на рівні 60%-65%. Після додатково проведеної оптимізації гіперпараметрів, що подаються на вхід моделі, отримали значення precision на рівні 62% та recall на рівні 71%. При цьому максимізуючою значення f1-score стала порогова ймовірність у розмірі 0,342 – тобто по цьому значенню надалі і приймалось рішення про приналежність до того чи іншого класу об’єктом.

Як наслідок отримали 12% неправильно виділених потенційних платників та 8% упущених клієнтів, що все ж таки стали платниками. Проте якщо все ж таки переходити до фінальної оцінки змодельованого доходу на основі всіх розглянутих вище підходів, отримали різницю у розмірі 1% між реальними та змодельованими даними в тестовій вибірці.

Отже, внаслідок виконання даної роботи були проаналізовані сучасні підходи та тенденції в маркетингу, процеси цілепокладання в ньому та підвищення його ефективності. В процесі моделювання використано підходи на основі когортного аналізу та методів машинного навчання, що дало змогу оцінити та порівняти їх результативність та дійти до найкращого рішення поставленого завдання. Надалі отримані результати можуть бути широко застосовувані в прийнятті рішень щодо покращення ефективності маркетингової діяльності компанії та продукту загалом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Валовий внутрішній продукт (2021). Мінфін. URL: https://index.minfin.com.ua/ua/economy/gdp/2021/.
2. На чому найбільше заробляє Google та яка структура доходів цієї компанії • Marketer. Marketer. URL: https://marketer.ua/ua/where-does-google-make-the-most-money/.
3. 27 lessons from Philip Kotler, the father of marketing. Postcron - Social Media Marketing Blog and Digital Marketing Blog. URL: https://postcron.com/en/blog/philip-kotler-advice-for-online-marketing/.
4. Visualizing the evolution of global advertising spend (1980-2020). Visual Capitalist. URL: https://www.visualcapitalist.com/evolution-global-advertising-spend-1980-2020/.
5. 5 marketing tips from John Jantsch. MBO Partners. URL: https://www.mbopartners.com/blog/how-market-small-business/five-marketing-tips-from-john-jantsch/.
6. What is performance marketing? [definition & examples]. CareerFoundry. URL: https://careerfoundry.com/en/blog/digital-marketing/performance-marketing/.
7. Pearce B. What is product marketing? - your complete guide. Product Marketing Alliance. URL: https://www.productmarketingalliance.com/what-is-product-marketing/.
8. What is brand marketing? A complete guide. Amazon Ads. URL: https://advertising.amazon.com/library/guides/brand-marketing.
9. Everything you need to know about unit economics. Accountancy Cloud. URL: https://theaccountancycloud.com/blogs/understanding-unit-economics.
10. What are unit economics and why are they important?. Lighter Capital. URL: https://www.lightercapital.com/blog/what-are-unit-economics.
11. A guide to unit economics for startups (2022 update) | fundsquire. Fundsquire Australia. URL: https://fundsquire.com.au/guide-unit-economics/.
12. ROI, ROMI, ROAS - що це і як розрахувати метрики. ITForce. URL: https://itforce.ua/blog/roi-romi-roas-chto-oboznachajut-metriki/.

13. What is A/B testing?. Kameleoon. URL: https://www.kameleoon.com/en/ab-testing.
14. Що таке когортний аналіз, чому він важливий для маркетингу і як його провести. Міжнародна Маркетингова Група. URL: https://www.marketing-ua.com/article/shho-take-kogortnij-analiz-chomu-vin-vazhlivij-dlya-marketingu-i-yak-jogo-provesti/.
15. Ставицький А. В. Методи згладжування. Офіційний сайт д.е.н., професора Ставицького А.В. URL: http://andriystav.cc.ua/Downloads/AppliedEco/03_Smoothing.pdf.
16. Ставицький А. В. Метричні методи класифікації та регресії. Алгоритм к найближчих сусідів. Офіційний сайт д.е.н., професора Ставицького А.В. URL: http://www.andriystav.cc.ua/Downloads/MITER/Lecture_06.pdf.
17. Sklearn.neighbors.KNeighborsClassifier. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.
18. Логістична регресія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Логістична_регресія.
19. Sklearn.linear_model.LogisticRegression. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
20. Sklearn.ensemble.RandomForestClassifier. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.
21. Breiman L. Random forests / Breiman L. // Machine Learning. – Vol. 45, N 1. – 2001. – P. 5–32.
22. Lightgbm.LGBMClassifier. LightGBM 3.3.2 documentation. URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html.
23. LightGBM (light gradient boosting machine). GeeksforGeeks. URL: https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/.

24. Чому DataScientist-и не використовують помилки першого та другого роду. Хабр. URL: https://habr.com/ru/articles/340048/.
25. How to use ROC curves and precision-recall curves for classification in python. MachineLearningMastery.com. URL: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.
26. Інтелектуальна обробка даних. Факультет комп'ютерних наук та кібернетики. URL: http://csc.knu.ua/media/filer_public/19/d5/19d56780-269a-4eef-bb3b-48ec8da23859/intelektualnaobrobkadanikh.pdf.
27. Numpy.corrcoef – numpy v1.24 manual. NumPy. URL: https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html.
28. Розподіл хі-квадрат. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Розподіл_хі-квадрат.
29. Sklearn.feature_selection.SelectKBest. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html.
30. Mwadulo M. W. A review on feature selection methods for classification tasks. International journal of computer applications technology and research. – Vol.5 No. 6. 2016. - P. 395–402.
31. Kohavi R., John G. H. Wrappers for feature subset selection. Artificial intelligence. 1997. Vol. 97, no. 1-2. P. 273–324.
32. Powerful feature selection with recursive feature elimination (RFE) of sklearn. Towards Data Science. URL: https://towardsdatascience.com/powerful-feature-selection-with-recursive-feature-elimination-rfe-of-sklearn-23efb2cdb54e.
33. Sklearn.feature_selection.RFE. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html.
34. A review of feature selection and its methods. Home | Sciendo. URL: https://sciendo.com/article/10.2478/cait-2019-0001.
35. Sklearn.feature_selection.SelectFromModel. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html#sklearn.feature_selection.SelectFromModel.

36. Ставицький А. В. Перевірка статистичних гіпотез. Офіційний сайт д.е.н., професора Ставицького А.В. URL: http://www.andriystav.cc.ua/Downloads/Econometrics/EViews/Lecture%2004.pdf.
37. Sklearn.preprocessing.StandardScaler. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.
38. Sklearn.preprocessing.OneHotEncoder. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.
39. Transforming categorical features to numerical features. CatBoost - open-source gradient boosting library. URL: https://catboost.ai/en/docs/concepts/algorithm-main-stages_cat-to-numberic.
40. Introducing Xverse! – A python package for feature selection and transformation. Towards Data Science. URL: https://towardsdatascience.com/introducing-xverse-a-python-package-for-feature-selection-and-transformation-17193cdcd067.
41. Sklearn.model_selection.GridSearchCV. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
42. Sklearn.model_selection.RandomizedSearchCV. scikit-learn. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.
43. S. Latif, "Customer annual income prediction using resampling approach", 2017 International Conference on Energy Communication Data Analytics and Soft Computing (ICECDS), pp. 3865-3870, 2017.
44. A machine learning based method for customer behavior prediction. Tehnicki vjesnik - technical gazette. 2019. Vol. 26, no. 6.

ДОДАТКИ

Додаток А

SQL-запит по формуванню датасету, необхідного для подальшого моделювання

```
with registrations as
         (select id client_id,
                 gender,
                 age,
                 country_code,
                 os_family,
                 reg_dt
          from marketing.v_users_registered
          where reg_dt >= '2023-02-04'
            and direction_id not in (30, 54)),
     costs as
         (select user_id,
                 coalesce(cpa_subchannel, 0::float) cac
          from aggregation.users_info
          where reg_date >= '2023-02-04'),
     pay_recurrents as
         (select a.*, discount percent_off
          from (select user_id,
                       root_order_id,
                       current_order_id
order_id,
                       current_dt,
                       scheduled_dt,
                       retry_attempt
retry_amount,
                       tariff_id,
                       case when retry_attempt is null then 1 else 0 end as
recurrent_status
                from payments.pay_schedule
                where current_dt >= '2023-02-04'
                  and root_order_id <> current_order_id) a
                   left join payments.v_tariff_logs_last tl on tl.tariff_id =
a.tariff_id),
     all_payments as
         (select order_id,
                 dt,
                 subscription_period_days
period,
                 gross_currency
currency_id,
                 card_bank
bank,
                 card_type
card,
                 card_country_id
card_country_code,
                 case when response_message ilike '%3d secure%' then 1 else 0
end was_3ds,
                 gross
root_gross,
                 descriptor
psp,
                 datediff('day', reg_dt, dt)
lt_user
          from aggregation.orders_paid_declined_pay
```

```
          where dt >= '2023-02-04'),
     insuff as
         (select user_id, count(*) insufficients
          from (select *, row_number() over (partition by order_id order by
notify_dt desc) as n
                from payments.orders_log
                where notify_dt >= '2023-02-04') a
          where response_code = '3.02'::varchar(4)
          group by 1),
     response as
         (select *
          from (select user_id, response_code last_response_code
                from payments.orders_log
                where notify_dt >= '2023-02-04'
                  and response_code is not null
                limit 1 over (partition by user_id order by notify_dt desc))
resp),
     declines as
         (select user_id, count(*) not_success_orders
          from (select *
                from payments.orders_log
                where notify_dt >= '2023-02-04'
                limit 1 over (partition by order_id order by notify_dt desc)) a
          where ((order_status_id <> 6) or (a.order_ps_status_id = 10) or
(a.order_ps_status_id = 11))
          group by 1),
     success as
         (select user_id, count(order_id) success_orders, sum(gross_usd) income
          from aggregation.orders_success_paid_pay
          where dt >= '2023-02-04'
          group by 1)

select registrations.client_id,
       pay_recurrents.scheduled_dt,
       pay_recurrents.root_order_id,
       registrations.gender,
       pay_1.lt_user,
       registrations.os_family,
       registrations.age,
       datediff('hour', registrations.reg_dt, pay_2.dt) hours_after_reg_buy,
       registrations.country_code,
       costs.cac,
       pay_recurrents.retry_amount,
       pay_1.period,
       success.income,
       pay_1.currency_id,
       pay_recurrents.percent_off,
       pay_1.psp,
       response.last_response_code,
       pay_1.bank,
       pay_1.was_3ds,
       pay_1.card,
       pay_1.card_country_code,
       success.success_orders,
       declines.not_success_orders,
       insuff.insufficients,
       pay_1.root_gross,
       (success.income - pay_1.root_gross)              not_root_gross,
       pay_recurrents.recurrent_status,
       pay_recurrents.current_dt
from registrations
         left join costs on costs.user_id = registrations.client_id
         join pay_recurrents on pay_recurrents.user_id = registrations.client_id
         join all_payments pay_1 on pay_1.order_id = pay_recurrents.order_id
```

```
        join all_payments pay_2 on pay_2.order_id =
pay_recurrents.root_order_id
        left join insuff on insuff.user_id = pay_recurrents.user_id
        left join response on response.user_id = pay_recurrents.user_id
        left join declines on declines.user_id = pay_recurrents.user_id
        left join success on success.user_id = pay_recurrents.user_id
```

Додаток Б

Огляд розподілів та залежностей у чисельних даних

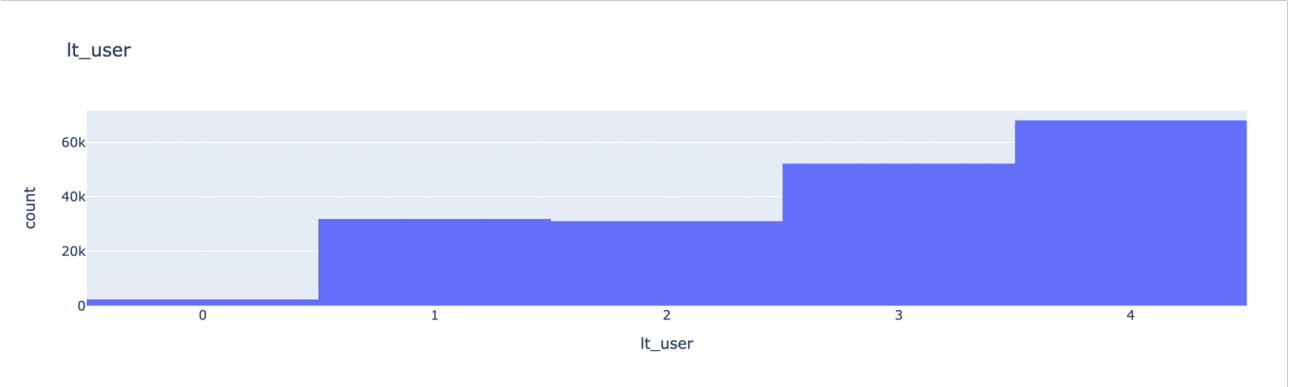


Рис.Б.1. Розподіл даних по лайфтаймах у клієнтів

Джерело: розрахунки автора.

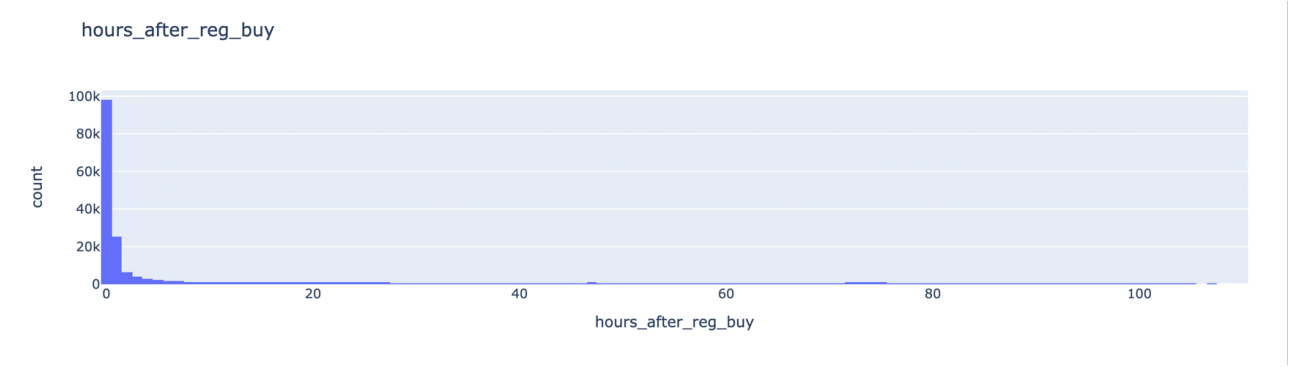


Рис.Б.2. Розподіл даних по годинах після реєстрації, коли клієнт здійснив покупку

Джерело: розрахунки автора.

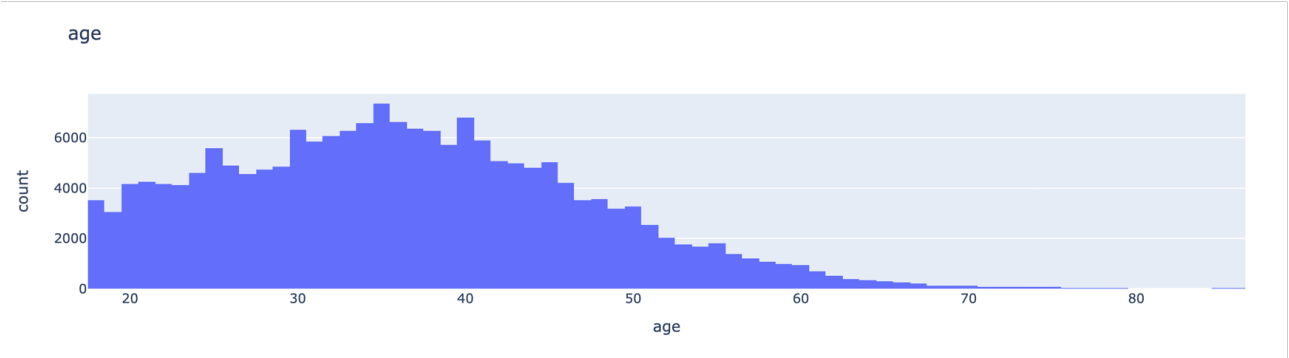


Рис.Б.3. Розподіл даних по віку клієнтів

Джерело: розрахунки автора.

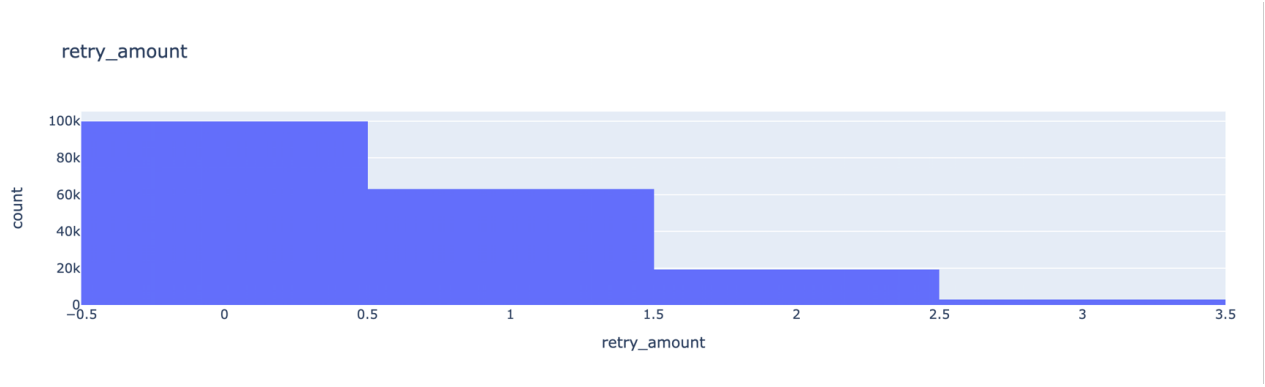


Рис.Б.4. Розподіл повторних спроб списання коштів

Джерело: розрахунки автора.

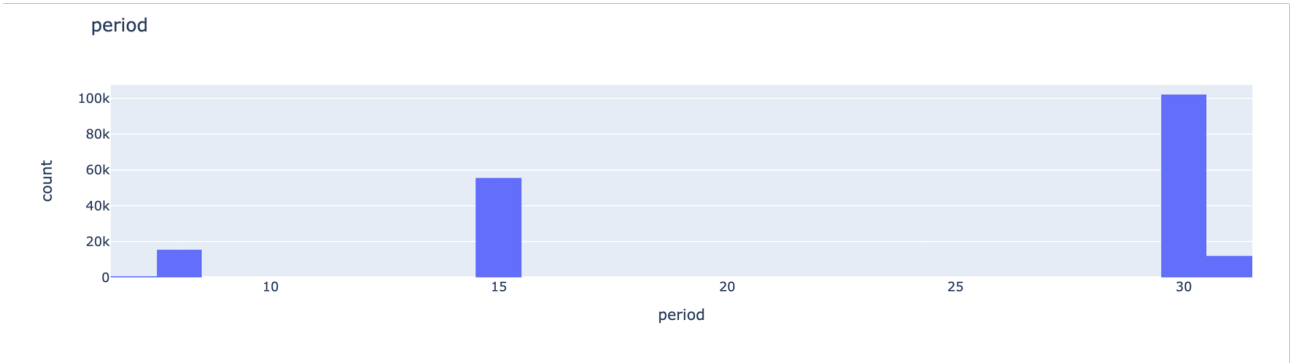


Рис.Б.5. Розподіл обраних періодів підписки

Джерело: розрахунки автора.

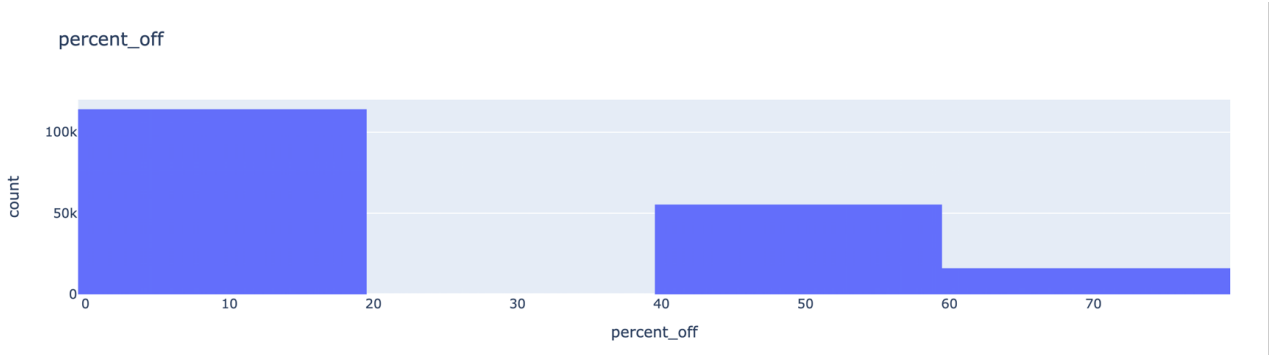


Рис.Б.6. Розподіл знижок, яку отримують клієнти при купівлі підписки

Джерело: розрахунки автора.

Рис.Б.7. Розподіл кількості успішних платежів

Джерело: розрахунки автора.

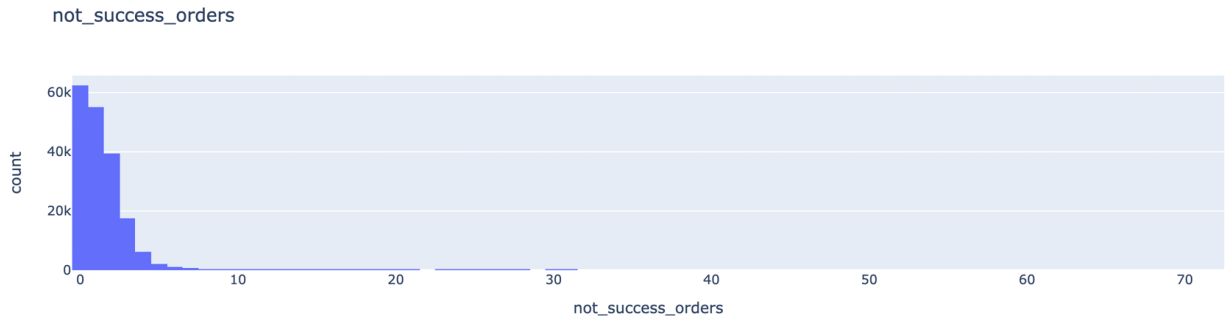


Рис.Б.8. Розподіл кількості невдалих платежів

Джерело: розрахунки автора.

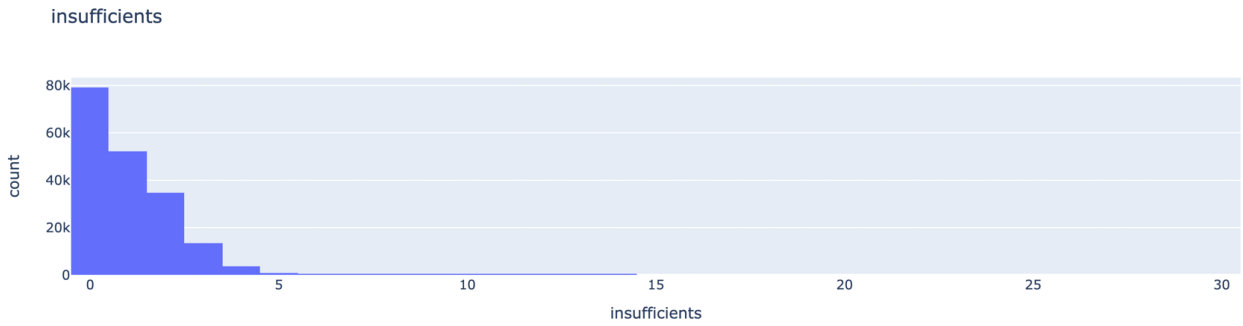


Рис.Б.9. Розподіл кількості невдалих платежів по причині недостатньої кількості грошей на рахунку

Джерело: розрахунки автора.

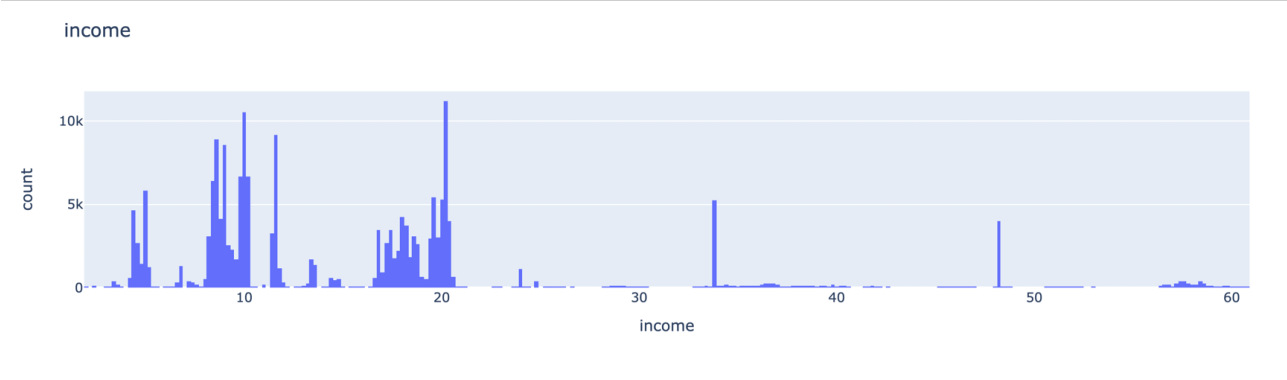


Рис.Б.10. Розподіл фактичних доходів від клієнтів

Джерело: розрахунки автора.

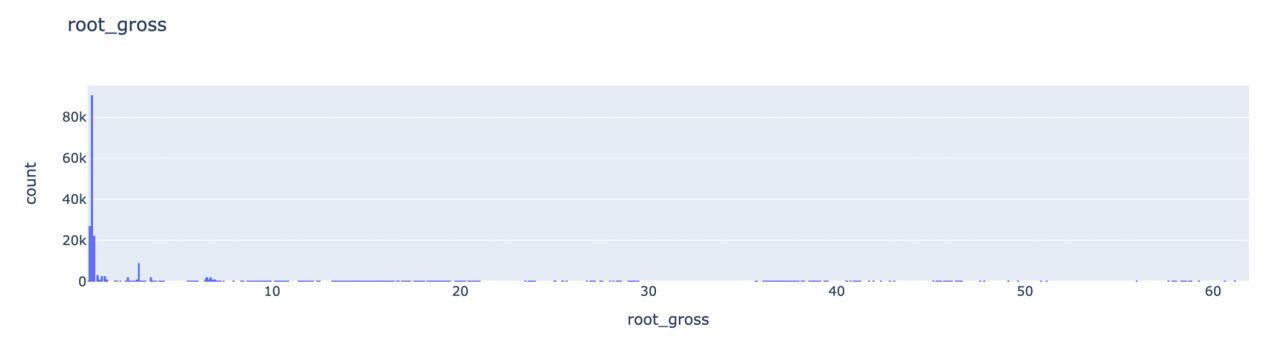


Рис.Б.11. Розподіл надходжень від купівлі підписок клієнтами

Джерело: розрахунки автора.

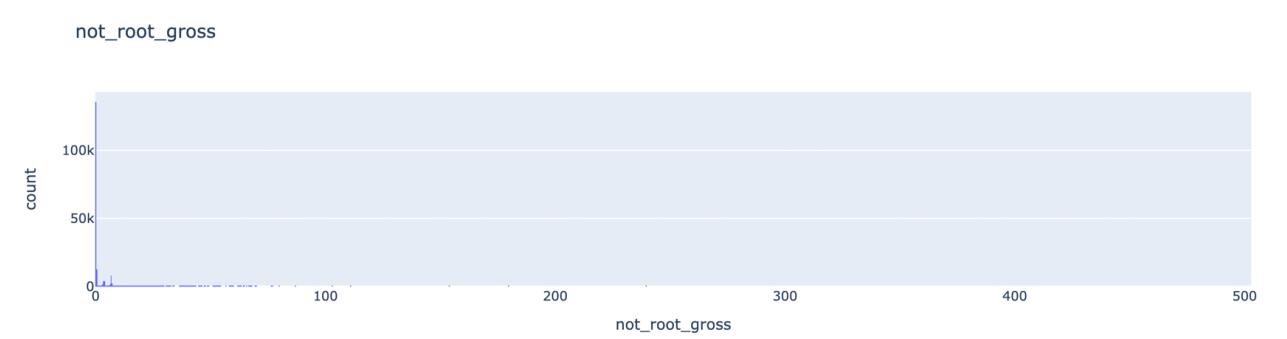


Рис.Б.12. Розподіл надходжень від відмінних від підписок опцій клієнтами

Джерело: розрахунки автора.

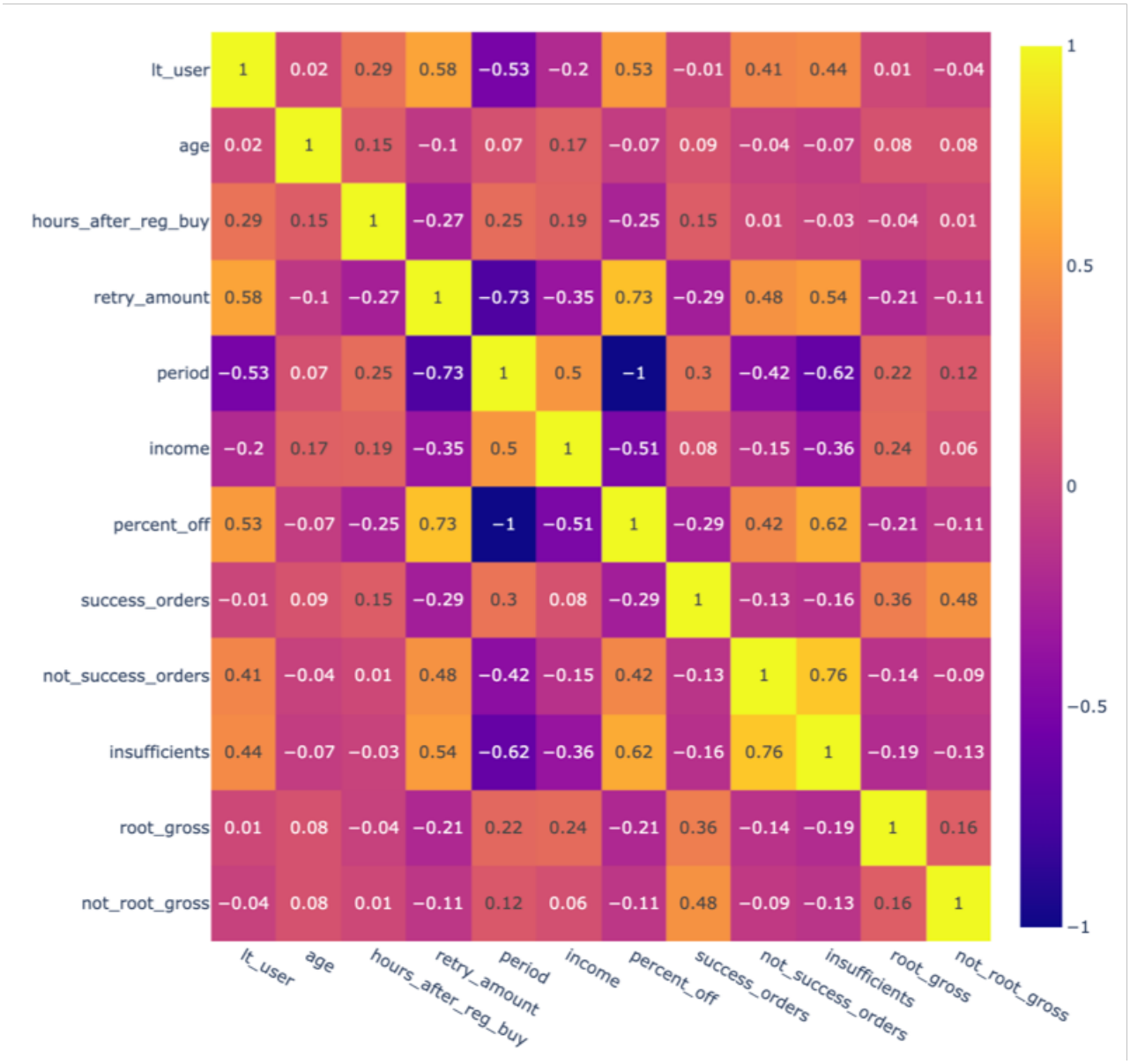


Рис.Б.13. Кореляційна матриця ознак

Джерело: розрахунки автора.

Продовження додатку Б

Рис.Б.14. Попарна залежність ознак та їх розсіювання

Джерело: розрахунки автора.

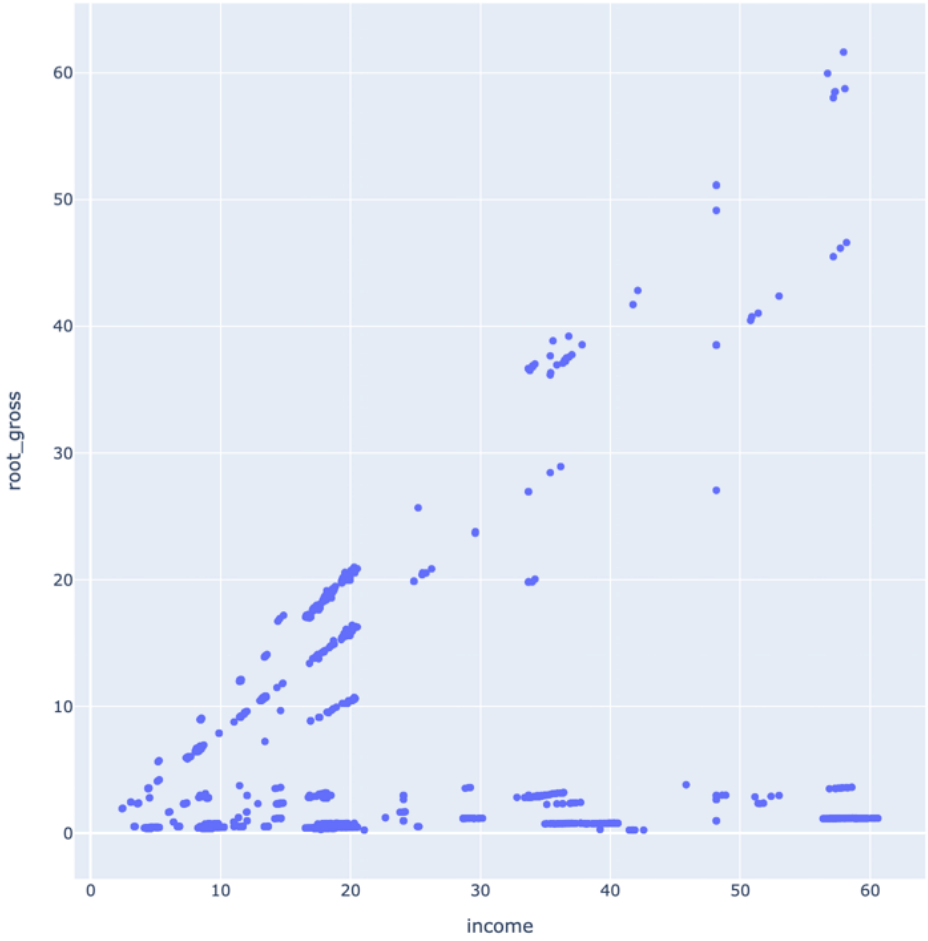


Рис.Б.15. Попарна залежність income та root_groos

Джерело: розрахунки автора.

Додаток В

Огляд розподілів та залежностей в категорійних, біноміальних даних та датах

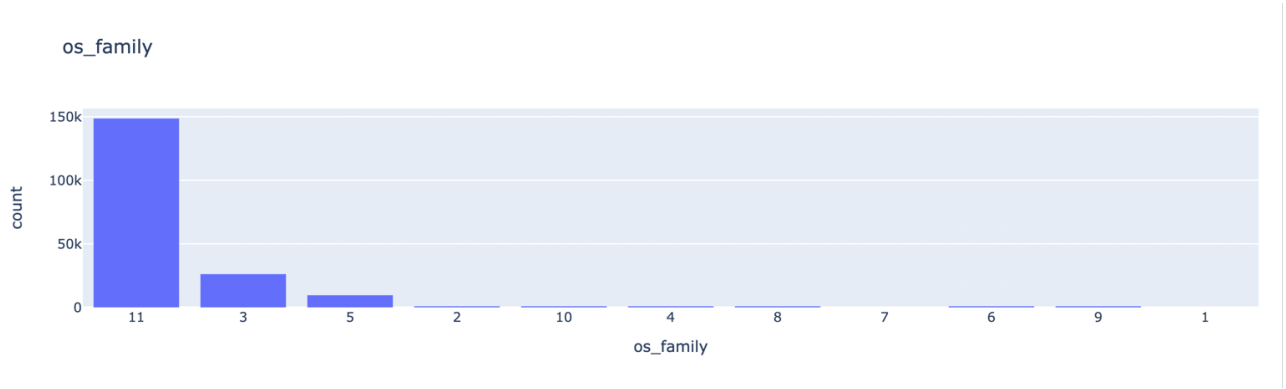


Рис.В.1. Розподіл операційних систем користувачів

Джерело: розрахунки автора.

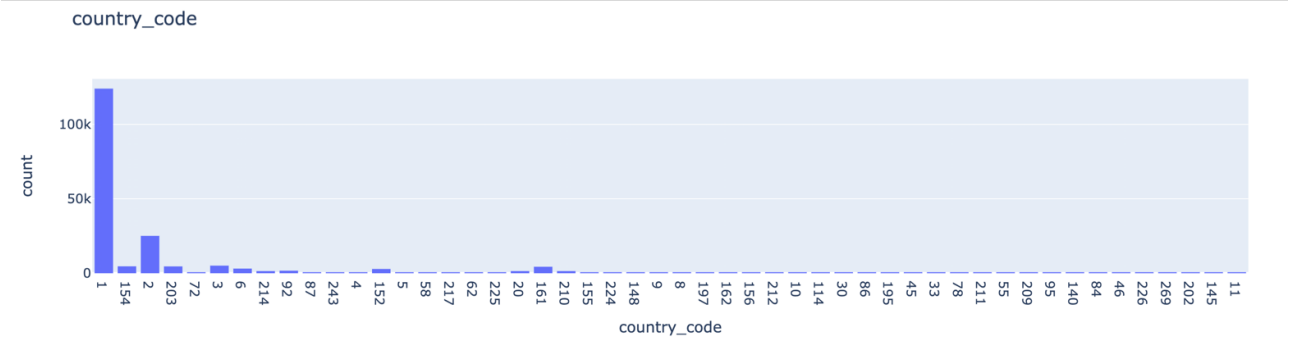


Рис.В.2. Розподіл країн клієнтів

Джерело: розрахунки автора.

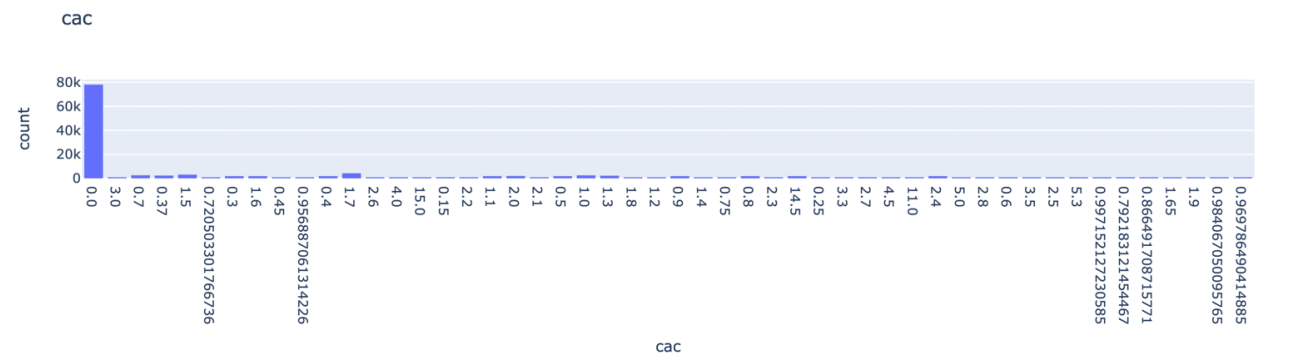


Рис.В.3. Розподіл вартості залучення клієнта

Джерело: розрахунки автора.

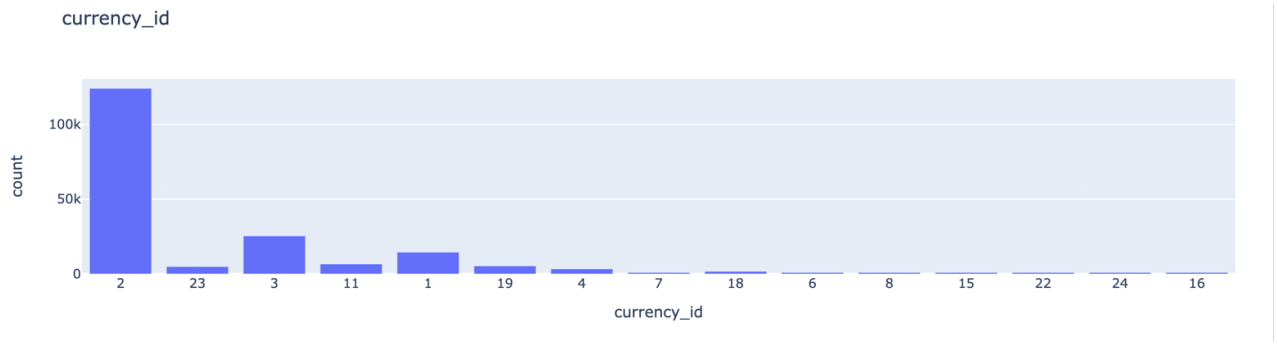


Рис.В.4. Розподіл валют клієнтів

Джерело: розрахунки автора.

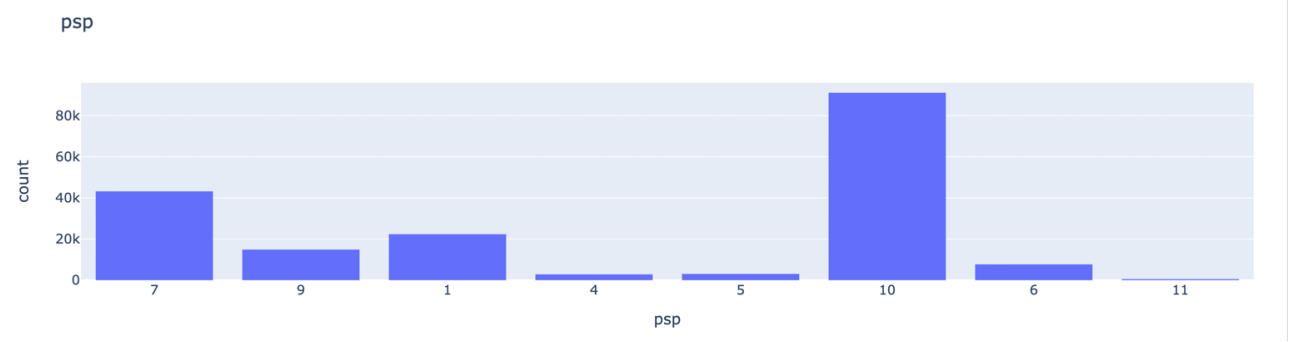


Рис.В.5. Розподіл провайдерів платежів

Джерело: розрахунки автора.

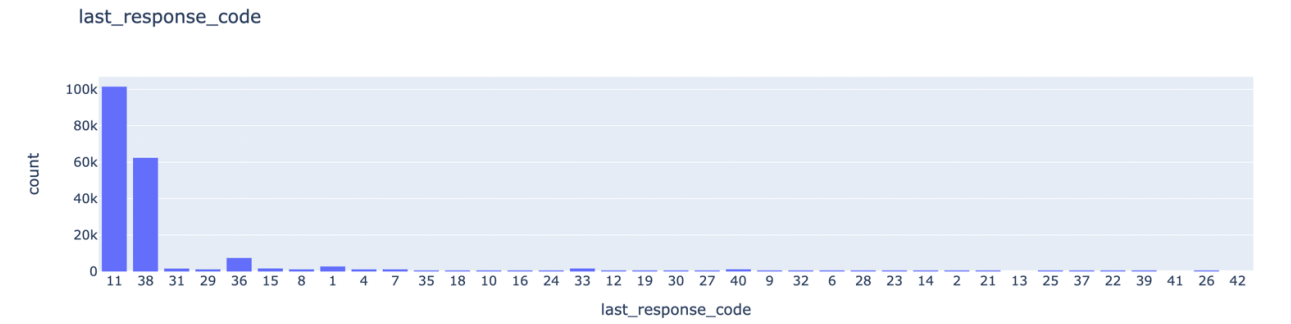


Рис.В.6. Розподіл відповідей, отриманих від банку при проведенні платежу

Джерело: розрахунки автора.

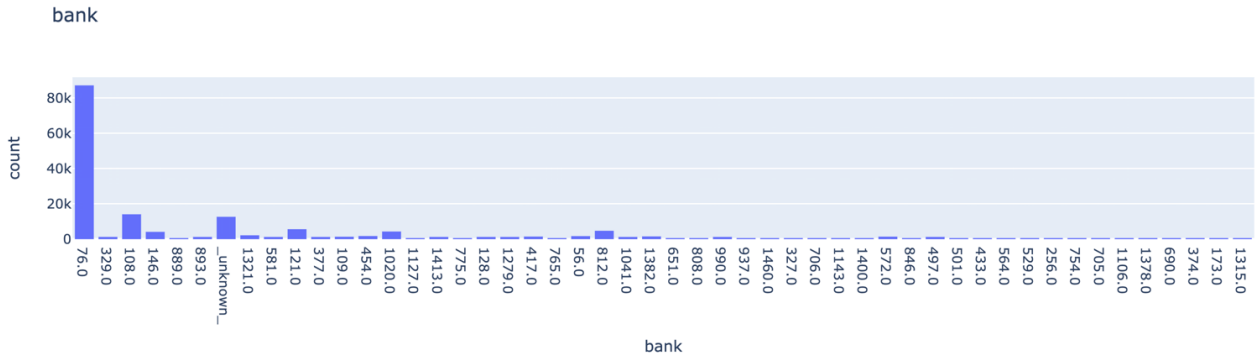


Рис.В.7. Розподіл банків клієнтів

Джерело: розрахунки автора.

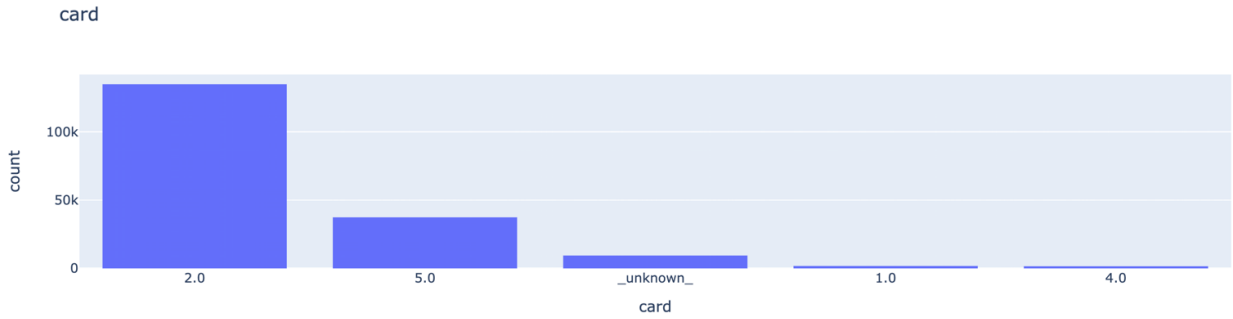


Рис.В.8. Розподіл видів карток клієнтів

Джерело: розрахунки автора.

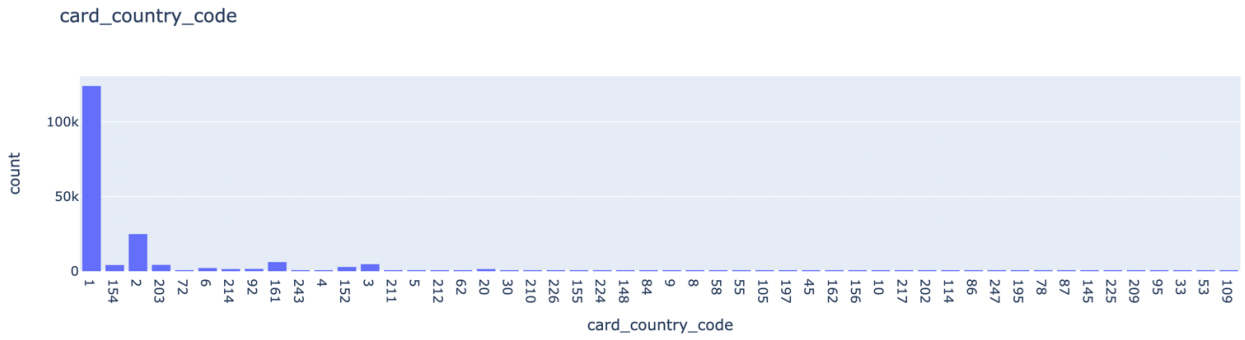


Рис.В.9. Розподіл країн оформлення карток

Джерело: розрахунки автора.

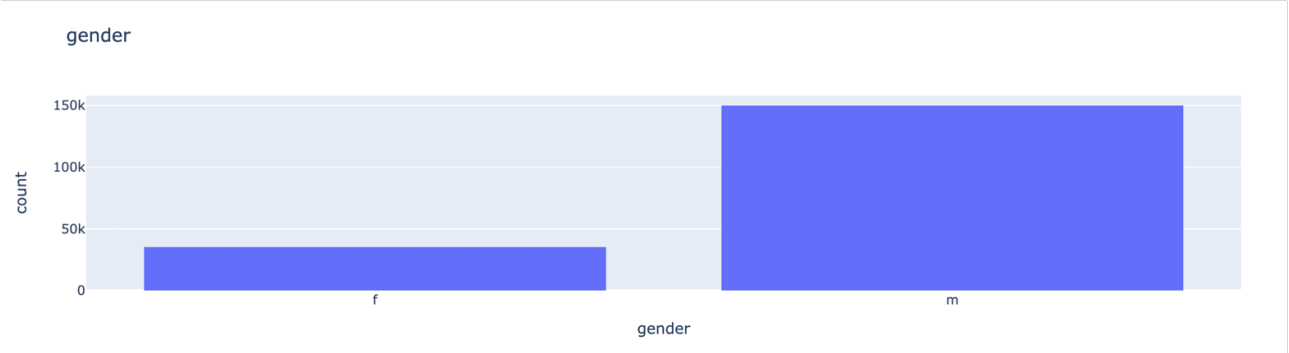


Рис.В.10. Розподіл статей клієнтів

Джерело: розрахунки автора.

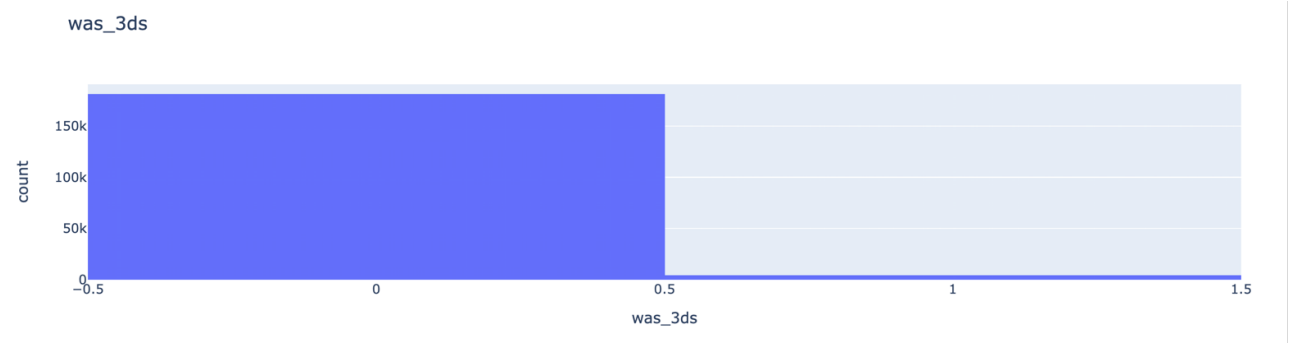


Рис.В.11. Розподіл індикаторів платежів з та без 3дс

Джерело: розрахунки автора.

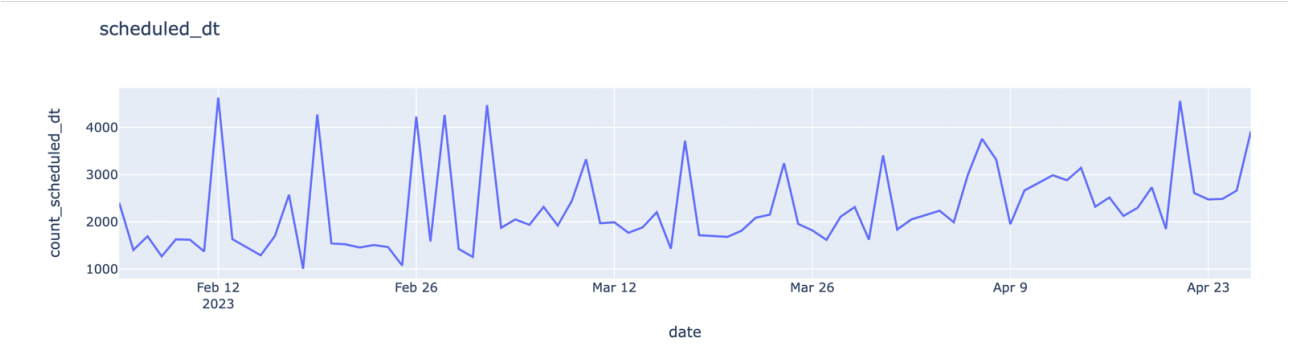


Рис.В.12. Розподіл дат запланованих списань коштів

Джерело: розрахунки автора.

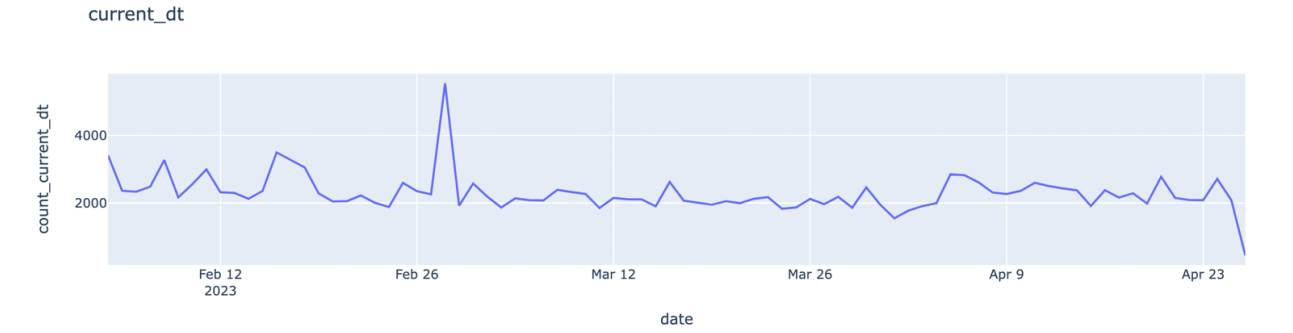


Рис.В.13. Розподіл поточних дат платежів

Джерело: розрахунки автора.

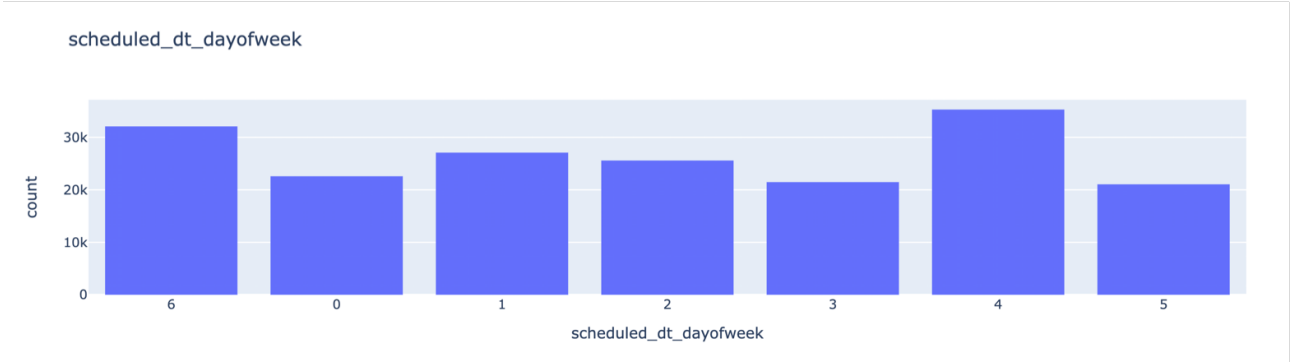


Рис.В.14. Розподіл днів тижня запланованих списань коштів

Джерело: розрахунки автора.

Додаток Г

Визначення найважливіших ознак при моделюванні

Таблиця Г.1

Коефіцієнти кореляції при визначенні важливості ознак за Пірсоном

Feature_name	Correlation
last_response_code	0.34
lt_user	-0.32
retry_amount	-0.29
insufficients	-0.28
not_success_orders	-0.27
period	0.24
income	0.13
bank	0.12
psp	0.07
scheduled_dt_dayofweek	0.07
cac	0.07
card_country_code	0.07
card	0.06
hours_after_reg_buy	-0.06
not_root_gross	0.06
country_code	0.06
age	0.06
root_gross	-0.06
gender	-0.04
currency_id	0.04
os_family	-0.03
was_3ds	-0.02
success_orders	0.01

Джерело: розрахунки автора.

Продовження додатку Г

Таблиця Г.2

Оцінка факторів при визначенні важливості ознак за методом Хі-квадрат

Feature_name	**Score**
retry_amount	3834.79
last_response_code	3725.85
lt_user	1870.60
period	1536.05
insufficients	517.08
income	361.72
not_success_orders	250.32
hours_after_reg_buy	243.06
root_gross	128.60
was_3ds	128.33
gender	69.31
bank	59.19
age	53.48
cac	48.75
not_root_gross	38.96
scheduled_dt_dayofweek	15.16
card	9.01
psp	7.00
card_country_code	6.40
country_code	4.78
success_orders	4.31
currency_id	1.64
os_family	0.35

Джерело: розрахунки автора.

Продовження додатку Г

Таблиця Г.3

Коефіцієнти, отримані з логістичної регресії при визначенні важливості ознак за методом-обгорткою RFE

Feature_name	**Coefficient**
root_gross	-6.20
not_root_gross	4.45
insufficients	-4.22
retry_amount	-2.74
last_response_code	2.70
bank	2.31
hours_after_reg_buy	-2.14
currency_id	-2.10
psp	2.04
country_code	1.98
not_success_orders	-1.67
scheduled_dt_dayofweek	-1.21
income	1.20
card	1.10
os_family	-0.94
age	0.84
period	-0.78
cac	0.61
card_country_code	0.45
was_3ds	-0.44
gender	-0.08
lt_user	-0.06
success_orders	0.01

Джерело: розрахунки автора.

Продовження додатку Г

Таблиця Г.4

Коефіцієнти, отримані під час побудови випадкового лісу при визначенні важливості ознак за вбудованим методом

Feature_name	**Coefficient**
last_response_code	0.10
lt_user	0.07
income	0.06
bank	0.06
psp	0.05
scheduled_dt_dayofweek	0.05
card	0.05
os_family	0.05
cac	0.05
card_country_code	0.05
currency_id	0.05
country_code	0.05
root_gross	0.04
age	0.04
hours_after_reg_buy	0.03
not_success_orders	0.03
not_root_gross	0.02
insufficients	0.02
retry_amount	0.02
period	0.01
success_orders	0.01
gender	0.01
was_3ds	0.00

Джерело: розрахунки автора.

Продовження додатку Г

Таблиця Г.5

Визначення найважливіших ознак методом voting на основі 4 різних підходів

Feature	Pearson	Chi-2	RFE	Random Forest	Total
scheduled_dt_dayofweek	True	True	True	True	4
root_gross	True	True	True	True	4
retry_amount	True	True	True	True	4
psp	True	True	True	True	4
period	True	True	True	True	4
not_success_orders	True	True	True	True	4
not_root_gross	True	True	True	True	4
last_response_code	True	True	True	True	4
insufficients	True	True	True	True	4
income	True	True	True	True	4
hours_after_reg_buy	True	True	True	True	4
country_code	True	True	True	True	4
card_country_code	True	True	True	True	4
card	True	True	True	True	4
cac	True	True	True	True	4
bank	True	True	True	True	4
age	True	True	True	True	4
lt_user	True	True	False	True	3
currency_id	True	False	True	True	3
was_3ds	False	True	True	False	2
os_family	False	False	True	True	2
gender	True	True	False	False	2
success_orders	False	False	False	False	0

Джерело: розрахунки автора.

Додаток Д

Python-код для реалізації моделювання доходу клієнтів

Libraries

```
import pandas as pd
import numpy as np
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
import pickle

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.metrics import (
    precision_score,
    recall_score,
    roc_auc_score,
    confusion_matrix,
    classification_report,
    roc_curve,
    precision_recall_curve,
    f1_score
)
from category_encoders.cat_boost import CatBoostEncoder

pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_rows', 100)
```

Import and fix

```
data = pd.read_csv('../02-data/dataset.csv')
```

Продовження додатку Д

```
feature_descr = pd.read_excel('../02-data/feature_descr.xlsx')
data.shape
feature_descr.shape
feature_descr.style.set_properties(**{'text-align': 'left'})
features_only = feature_descr.loc[~(feature_descr['name'].isin(['recurrent_status']))]
features_only['type'].value_counts()
num_fs = features_only.loc[(feature_descr['type'] == 'NUM'), 'name'].tolist()
cat_fs = features_only.loc[(feature_descr['type'] == 'CAT'), 'name'].tolist()
date_fs = features_only.loc[(feature_descr['type'] == 'DATE'), 'name'].tolist()
bool_fs = features_only.loc[(feature_descr['type'] == 'BOOL'), 'name'].tolist()

is_na_table = data.isna().sum(axis=0)
is_na_table[is_na_table > 0]
data['bank'] = np.where(data['bank'].isna(), '_unknown_', data['bank'])
data['card'] = np.where(data['card'].isna(), '_unknown_', data['card'])
is_na_table = data.isna().sum(axis=0)
is_na_table[is_na_table > 0]
```

Data types

```
data[cat_fs].dtypes
for fs in cat_fs:
    data[fs] = data[fs].astype(str)
data[cat_fs].dtypes
```

Data visualization

```
print('Count samples: ', data.shape[0], '\nCount users: ',
data['client_id'].unique().shape[0])
```

Продовження додатку Д

```
data['client_id'].value_counts().to_frame('count')['count'].value_counts(normalize=True)
data['root_order_id'].unique().shape[0]
(data['root_order_id'].value_counts().to_frame('count')['count']).value_counts(normalize=True)
```

User data

```
data_client = data.sort_values('scheduled_dt').groupby('client_id').tail(1)
for fs in num_fs:
    data_client[fs] = data_client[fs]
    fig = px.histogram(data_client, x=fs, title=fs)
    fig.show()
```

Correlation

```
fig = px.imshow(data_client[num_fs].corr() ,text_auto=True, aspect="auto")
fig.show()
```

Pairplot

```
%matplotlib inline
sns.pairplot(data_client[num_fs], kind='scatter', diag_kind='hist')
plt.show()
fig = px.scatter(data_client[num_fs].sample(10_000), x='income', y='root_gross', height=800, width=800)
fig.show()
```

Data overview

```
for fs in cat_fs:
    top_n = data_client[fs].value_counts().iloc[:50].index
    fig = px.histogram(data_client[data_client[fs].isin(top_n)], x=fs, title=fs)
```

Продовження додатку Д

```
    fig.show()

for fs in bool_fs:
    data_client[fs] = data_client[fs]
    fig = px.histogram(data_client, x=fs, title=fs)
    fig.show()

for fs in date_fs:
    new_name = 'count_' + fs
    to_viz = pd.to_datetime(data_client[fs]).dt.date.value_counts().to_frame(new_name).reset_index().rename(columns={'index':'date'})
    fig = px.line(to_viz.sort_values(by='date'), x='date', y=new_name, title=fs)
    fig.show()

data_client['scheduled_dt_dayofweek'] = pd.to_datetime(data_client['scheduled_dt']).dt.dayofweek.astype(str)

fig = px.histogram(data_client, x='scheduled_dt_dayofweek', title='scheduled_dt_dayofweek')
fig.show()
```

Preparation

```
num_fs_sel = list(set(num_fs) - set(['percent_off']))
cat_fs_sel = cat_fs + ['scheduled_dt_dayofweek']
bool_fs_sel = bool_fs
fs_sel = num_fs_sel + cat_fs_sel + bool_fs_sel
```

Split

Продовження додатку Д

```
data_client.sort_values(by='scheduled_dt', inplace=True)
X = data_client[fs_sel]
y = data_client['recurrent_status']
test_size = 0.8

n = int(X.shape[0]*test_size)

X_train = X[:n]
X_test = X[n:]

y_train = y[:n]
y_test = y[n:]

y_train_regr = y[:n] * X[:n]['income']
y_test_regr = y[n:] * X[n:]['income']

X_train.shape, X_test.shape
y_train.mean(), y_test.mean()
y_train_regr.mean(), y_test_regr.mean()
```

Feature processing

```
scaler = StandardScaler()
X_train_num_scal = pd.DataFrame(scaler.fit_transform(X_train[num_fs_sel]),
index=X_train.index, columns=num_fs_sel)
X_test_num_scal = pd.DataFrame(scaler.transform(X_test[num_fs_sel]),
index=X_test.index, columns=num_fs_sel)
X_train_bool = X_train[bool_fs_sel].copy()
X_train_bool['gender'] = X_train_bool['gender'].map(dict(m=1, f=0))
```

Продовження додатку Д

```
X_test_bool = X_test[bool_fs_sel].copy()
X_test_bool['gender'] = X_test_bool['gender'].map(dict(m=1, f=0))

enc_ohe = OneHotEncoder(min_frequency=0.01, handle_unknown='infrequent_if_exist')
X_train_cat_ohe = enc_ohe.fit_transform(X_train[cat_fs_sel])
X_train_cat_ohe = pd.DataFrame(X_train_cat_ohe.todense(), columns=enc_ohe.get_feature_names_out(cat_fs_sel), index=X_train.index)

X_test_cat_ohe = pd.DataFrame(enc_ohe.transform(X_test[cat_fs_sel]).todense(), columns=enc_ohe.get_feature_names_out(cat_fs_sel), index=X_test.index)

X_train_cat_ohe.shape
X_test_cat_ohe.shape
enc_cat = CatBoostEncoder()
X_train_cat_cbe = enc_cat.fit_transform(X_train[cat_fs_sel], y_train)
X_test_cat_cbe = enc_cat.transform(X_test[cat_fs_sel])

X_train_final_ohe = pd.concat([X_train_num_scal, X_train_cat_ohe, X_train_bool], axis=1)
X_train_final_cat = pd.concat([X_train_num_scal, X_train_cat_cbe, X_train_bool], axis=1)

X_test_final_ohe = pd.concat([X_test_num_scal, X_test_cat_ohe, X_test_bool], axis=1)
X_test_final_cat = pd.concat([X_test_num_scal, X_test_cat_cbe, X_test_bool], axis=1)

X_train_final_ohe.shape, X_test_final_ohe.shape
```

Продовження додатку Д

```
X_train_final_cat.shape, X_test_final_cat.shape
```

Important factors

```
n = int(X_train_final_cat.shape[1]*0.9)
```

Pearson correlation

```
def cor_selector(X, y, n):
    feature_name = X.columns
    cor_list = []
    # calculate the correlation with y for each feature
    for i in X.columns.tolist():
        cor = np.corrcoef(X[i], y)[0, 1]
        cor_list.append(cor)

    lists=np.abs(cor_list)
    prep=sorted(lists)
    ind=X.iloc[:,np.argsort(np.abs(cor_list))].columns.tolist()
    cor_feature = X.iloc[:,np.argsort(np.abs(cor_list))[-n:]].columns.tolist()
    cor_support = [True if i in cor_feature else False for i in feature_name]

    matrix = [[feature_name[i], cor_list[i]] for i in range(len(feature_name))]
    matrix.sort(key=lambda x: abs(x[1]), reverse=True)
    for row in matrix:
        print(row[0], '-', row[1])

    return cor_support, cor_feature, prep, ind
cor_support, cor_feature, prep, ind = cor_selector(X_train_final_cat, y_train, n)
print(cor_feature)
print(str(len(cor_feature)), 'selected features')
```

Продовження додатку Д

Chi-2

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
X_train_norm = MinMaxScaler().fit_transform(X_train_final_cat)
chi_selector = SelectKBest(chi2, k=n)
chi_selector.fit(X_train_norm, y_train)
chi_support = chi_selector.get_support()
chi_feature = X_train_final_cat.loc[:,chi_support].columns.tolist()
print(chi_feature)
print(str(len(chi_feature)), 'selected features')
feature_names = X_train_final_cat.columns
scores = chi_selector.scores_
p_val = chi_selector.pvalues_
feature_scores = pd.DataFrame({'Feature Name': feature_names, 'Score': scores, 'P_val': p_val})
sorted_df = feature_scores.sort_values(by='Score', ascending=False)
sorted_df
```

Wrapper

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
rfe_selector = RFE(estimator=LogisticRegression(solver='liblinear'), n_features_to_select=n, step=1)
rfe_selector.fit(X_train_norm, y_train)
rfe_support = rfe_selector.get_support()
rfe_feature = X_train_final_cat.loc[:,rfe_support].columns.tolist()
feature_coefs = pd.DataFrame({
   'Feature': rfe_feature,
```

Продовження додатку Д

```
'Coefficient': rfe_selector.estimator_.coef_[0]
   })
sorted_df = feature_coefs.sort_values(by='Coefficient', key=abs, ascending=False)
sorted_df
print(rfe_feature)
print(str(len(rfe_feature)), 'selected features')
```

Embeded

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
embeded_rf_selector = SelectFromModel(RandomForestClassifier(), max_features=n,
threshold=-np.inf)
embeded_rf_selector.fit(X_train_final_cat, y_train)
embeded_rf_support = embeded_rf_selector.get_support()
embeded_rf_feature = X_train_final_cat.loc[:,embeded_rf_support].columns.tolist()
feature_coefs = pd.DataFrame({
   'Feature': X_train_final_cat.columns,
   'Importance': embeded_rf_selector.estimator_.feature_importances_
   })
sorted_df = feature_coefs.sort_values(by='Importance', ascending=False)
sorted_df
print(embeded_rf_feature)
print(str(len(embeded_rf_feature)), 'selected features')
```

Voting

```
feature_selection_df = pd.DataFrame({'Feature': X_train_final_cat.columns, 'Pearson':
cor_support, 'Chi-2': chi_support, 'RFE': rfe_support,
                                    'Random Forest': embeded_rf_support})
# count the selected times for each feature
```

Продовження додатку Д

```
feature_selection_df.loc[:, 'Total'] = np.sum(feature_selection_df[feature_selection_df.columns.difference(['Feature'])], axis=1)

feature_selection_df = feature_selection_df.sort_values(['Total','Feature'] , ascending=False)
feature_selection_df.reset_index(drop=True, inplace=True)
feature_selection_df
top_features_1=feature_selection_df[feature_selection_df['Total'] >= 4]
top1_features=top_features_1['Feature'].tolist()
print(top1_features)
top_features_2=feature_selection_df[feature_selection_df['Total'] >= 3]
top2_features=top_features_2['Feature'].tolist()
print(top2_features)
```

Modeling

```
# Function for comparing train/test metrics

def calc_metrics(proba_train, proba_test, y_train, y_test):
    precision, recall, thresholds = precision_recall_curve(y_train, proba_train)
    # convert to f score
    fscore = np.where(precision + recall != 0, (2 * precision * recall) / np.where(precision + recall == 0, 1, precision + recall), 0)
    # locate the index of the largest f score
    ix = np.argmax(fscore)
    print('Best Threshold = ', round(thresholds[ix], 3))

    res = [
        dict(
```

```
            split='Train',
            roc_auc=roc_auc_score(y_train, proba_train > thresholds[ix]),
            precision=precision_score(y_train, proba_train > thresholds[ix]),
            recall=recall_score(y_train, proba_train > thresholds[ix]),
            f1_score=f1_score(y_train, proba_train > thresholds[ix]),
            y_perc=y_train.mean(),
            y_hat_perc=(proba_train > thresholds[ix]).mean()
        ),
        dict(
            split='Test',
            roc_auc=roc_auc_score(y_test, proba_test > thresholds[ix]),
            precision=precision_score(y_test, proba_test > thresholds[ix]),
            recall=recall_score(y_test, proba_test > thresholds[ix]),
            f1_score=f1_score(y_test, proba_test > thresholds[ix]),
            y_perc=y_test.mean(),
            y_hat_perc=(proba_test > thresholds[ix]).mean()
        )
    ]

    return pd.DataFrame(res)
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

knn.fit(X_train_final_ohe, y_train)
calc_metrics(
    knn.predict_proba(X_train_final_ohe)[:, 1],
    knn.predict_proba(X_test_final_ohe)[:, 1],
```

Продовження додатку Д

```
    y_train,
    y_test)

knn.fit(X_train_final_cat[top2_features], y_train)
calc_metrics(
    knn.predict_proba(X_train_final_cat[top2_features])[:, 1],
    knn.predict_proba(X_test_final_cat[top2_features])[:, 1],
    y_train,
    y_test)

knn.fit(X_train_final_cat[top1_features], y_train)
calc_metrics(
    knn.predict_proba(X_train_final_cat[top1_features])[:, 1],
    knn.predict_proba(X_test_final_cat[top1_features])[:, 1],
    y_train,
    y_test)
```

Logistic Regression

```
clf = LogisticRegression(solver='liblinear')

clf.fit(X_train_final_ohe, y_train)
calc_metrics(
    clf.predict_proba(X_train_final_ohe)[:, 1],
    clf.predict_proba(X_test_final_ohe)[:, 1],
    y_train,
    y_test

clf.fit(X_train_final_cat[top2_features], y_train)
calc_metrics(
```

Продовження додатку Д

```
    clf.predict_proba(X_train_final_cat[top2_features])[:, 1],
    clf.predict_proba(X_test_final_cat[top2_features])[:, 1],
    y_train,
    y_test)

clf.fit(X_train_final_cat[top1_features], y_train)
calc_metrics(
    clf.predict_proba(X_train_final_cat[top1_features])[:, 1],
    clf.predict_proba(X_test_final_cat[top1_features])[:, 1],
    y_train,
    y_test)
```

Random Forest Classifier

```
rfm=RandomForestClassifier()

rfm.fit(X_train_final_ohe, y_train)
calc_metrics(
    rfm.predict_proba(X_train_final_ohe)[:, 1],
    rfm.predict_proba(X_test_final_ohe)[:, 1],
    y_train,
    y_test)

rfm=RandomForestClassifier(max_depth=10)
rfm.fit(X_train_final_ohe, y_train)
calc_metrics(
    rfm.predict_proba(X_train_final_ohe)[:, 1],
    rfm.predict_proba(X_test_final_ohe)[:, 1],
    y_train,
    y_test)
```

Продовження додатку Д

```
rfm.fit(X_train_final_cat[top2_features], y_train)
calc_metrics(
   rfm.predict_proba(X_train_final_cat[top2_features])[:, 1],
   rfm.predict_proba(X_test_final_cat[top2_features])[:, 1],
   y_train,
   y_test)

rfm.fit(X_train_final_cat[top1_features], y_train)
calc_metrics(
   rfm.predict_proba(X_train_final_cat[top1_features])[:, 1],
   rfm.predict_proba(X_test_final_cat[top1_features])[:, 1],
   y_train,
   y_test)
```

LGBMClassifier

```
from lightgbm import LGBMClassifier
lgb_m = LGBMClassifier()

lgb_m.fit(X_train_final_ohe, y_train)
calc_metrics(
   lgb_m.predict_proba(X_train_final_ohe)[:, 1],
   lgb_m.predict_proba(X_test_final_ohe)[:, 1],
   y_train,
   y_test)

lgb_m.fit(X_train_final_cat[top2_features], y_train)
calc_metrics(
   lgb_m.predict_proba(X_train_final_cat[top2_features])[:, 1],
   lgb_m.predict_proba(X_test_final_cat[top2_features])[:, 1],
```

Продовження додатку Д

```
    y_train,
    y_test)

lgb_m.fit(X_train_final_cat[top1_features], y_train)
calc_metrics(
    lgb_m.predict_proba(X_train_final_cat[top1_features])[:, 1],
    lgb_m.predict_proba(X_test_final_cat[top1_features])[:, 1],
    y_train,
    y_test)

from sklearn.model_selection import RandomizedSearchCV
param_grid = {
    'n_estimators': np.arange(0, 2000, 50),
    'learning_rate': np.arange(0.005, 0.5, 0.005),
    'max_depth': np.arange(5, 300, 3),
    'min_child_samples': np.arange(10, 500, 10)
}

lgbm_random = RandomizedSearchCV(
    estimator= LGBMClassifier(),
    param_distributions=param_grid,
    n_iter = 50,
    scoring='f1',
    cv = 6,
    verbose=1,
    random_state=42,
    n_jobs=-1,
    return_train_score=True
)
```

Продовження додатку Д

```
%%time
lgbm_random.fit(X_train_final_cat[top1_features], y_train)
print(lgbm_random.best_params_)

lgb_m_4 = LGBMClassifier(**lgbm_random.best_params_)
lgb_m_4.fit(X_train_final_cat[top1_features], y_train)
calc_metrics(
    lgb_m_4.predict_proba(X_train_final_cat[top1_features])[:, 1],
    lgb_m_4.predict_proba(X_test_final_cat[top1_features])[:, 1],
    y_train,
    y_test)
```

Classification Detailed

```
# Function for threshold

def thresh(proba_train, y_train):
    precision, recall, thresholds = precision_recall_curve(y_train, proba_train)
    # convert to f score
    fscore = np.where(precision + recall != 0, (2 * precision * recall) /
np.where(precision + recall == 0, 1, precision + recall), 0)
    # locate the index of the largest f score
    ix = np.argmax(fscore)
    threshold=round(thresholds[ix], 3)

    return threshold, ix,  fscore

pay_prob_train=lgb_m.predict_proba(X_train_final_cat[top1_features])[:, 1]
pay_prob_test=lgb_m.predict_proba(X_test_final_cat[top1_features])[:, 1]
```

Продовження додатку Д

```
lgb_m.fit(X_train_final_cat[top1_features], y_train)
threshold, ix, fscore=thresh(
    lgb_m.predict_proba(X_train_final_cat[top1_features])[:, 1],
    y_train)
print (threshold, ix, fscore)

precision, recall, thresholds = precision_recall_curve(y_test, pay_prob_test)
precision_sc=precision_score(y_test, pay_prob_test > threshold)
recall_sc=recall_score(y_test, pay_prob_test > threshold)

plt.figure(figsize=(8, 6))
plt.plot(thresholds, precision[:-1], label='Precision', color='purple')
plt.plot(thresholds, recall[:-1], label='Recall', color='plum')
plt.scatter(threshold, precision_sc, c='red', marker='o', label='Best Threshold')
plt.scatter(threshold, recall_sc, c='red', marker='o')
plt.xlabel('Threshold')
plt.ylabel('Precision/Recall')
plt.title('Precision-Recall Curve')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label='Precision-Recall Curve', color='purple')
plt.scatter([recall_sc], [precision_sc], c='red', marker='o', label='Best Threshold')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
plt.show()
```

Продовження додатку Д

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, pay_prob_test)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='purple', label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random')
plt.scatter(fpr[np.argmax(tpr >= threshold)], tpr[np.argmax(tpr >= threshold)], c='red',
marker='o', label='Best Threshold')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC-curve')
plt.legend(loc="lower right")
plt.show()

from sklearn.metrics import confusion_matrix
import seaborn as sns
proba_test = lgb_m.predict_proba(X_test_final_cat[top1_features])[:, 1]
y_pred = (proba_test > threshold).astype(int)
cm = confusion_matrix(y_test, y_pred)
labels = ['Negative', 'Positive']
sns.heatmap(cm, annot=True, fmt='d', cmap='PuRd', xticklabels=labels,
yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Продовження додатку Д

```
import lightgbm as lgb
lgb.plot_importance(lgb_m, height=0.5, color='purple')
plt.show()
```

Finish

```
pay_prob_train=lgb_m.predict_proba(X_train_final_cat[top1_features])[:, 1]
pay_prob_test=lgb_m.predict_proba(X_test_final_cat[top1_features])[:, 1]

threshold=thresh(
    lgb_m.predict_proba(X_train_final_cat[top1_features])[:, 1],
    y_train)

prediction_train = np.where(pay_prob_train >= threshold, 1, 0)
prediction_test = np.where(pay_prob_test >= threshold, 1, 0)

test_size = 0.8
n = int(X.shape[0]*test_size)

gross_train_2=prediction_train.sum()
gross_test_2=prediction_test.sum()

print(gross_train_2, y_train.sum())
print(gross_test_2, y_test.sum())



train_actual = y_train.sum()
train_predicted = gross_train_2.sum()

test_actual = y_test.sum()
```

Продовження додатку Д

```
test_predicted = gross_test_2.sum()

labels = ['Actual', 'Predicted']
values_train = [train_actual, train_predicted]
values_test = [test_actual, test_predicted]

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

ax1.bar(labels, values_train, color=['purple', 'plum'])
ax1.set_title('Train Set')
ax1.set_ylabel('Total Gross Income')

for i, v in enumerate(values_train):
    ax1.text(i, v, str(round(v, 2)), ha='center', va='bottom')

ax2.bar(labels, values_test, color=['purple', 'plum'])
ax2.set_title('Test Set')
ax2.set_ylabel('Total Gross Income')

for i, v in enumerate(values_test):
    ax2.text(i, v, str(round(v, 2)), ha='center', va='bottom')

plt.show()

test_size = 0.8

n = int(X.shape[0]*test_size)

gross_train_1=pay_prob_train* X[:n]['income']
```

Продовження додатку Д

```
gross_test_1=pay_prob_test* X[n:]['income']
gross_train_2=prediction_train.sum()* X[:n]['income'].mean()
gross_test_2=prediction_test.sum()* X[n:]['income'].mean()

print(gross_train_1.sum(), y_train_regr.sum())
print(gross_train_2, y_train_regr.sum())
print(gross_test_1.sum(), y_test_regr.sum())
print(gross_test_2, y_test_regr.sum())



train_actual = y_train_regr.sum()
train_predicted = gross_train_2.sum()

test_actual = y_test_regr.sum()
test_predicted = gross_test_2.sum()

labels = ['Actual', 'Predicted']
values_train = [train_actual, train_predicted]
values_test = [test_actual, test_predicted]

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

ax1.bar(labels, values_train, color=['purple', 'plum'])
ax1.set_title('Train Set')
ax1.set_ylabel('Total Gross Income')

for i, v in enumerate(values_train):
    ax1.text(i, v, str(round(v, 2)), ha='center', va='bottom')
```

Продовження додатку Д

```
ax2.bar(labels, values_test, color=['purple', 'plum'])
ax2.set_title('Test Set')
ax2.set_ylabel('Total Gross Income')

for i, v in enumerate(values_test):
    ax2.text(i, v, str(round(v, 2)), ha='center', va='bottom')

plt.show()
```